Database Principles Crash Course

Module Overview

- Database Concepts
- Fundamentals About Transactions

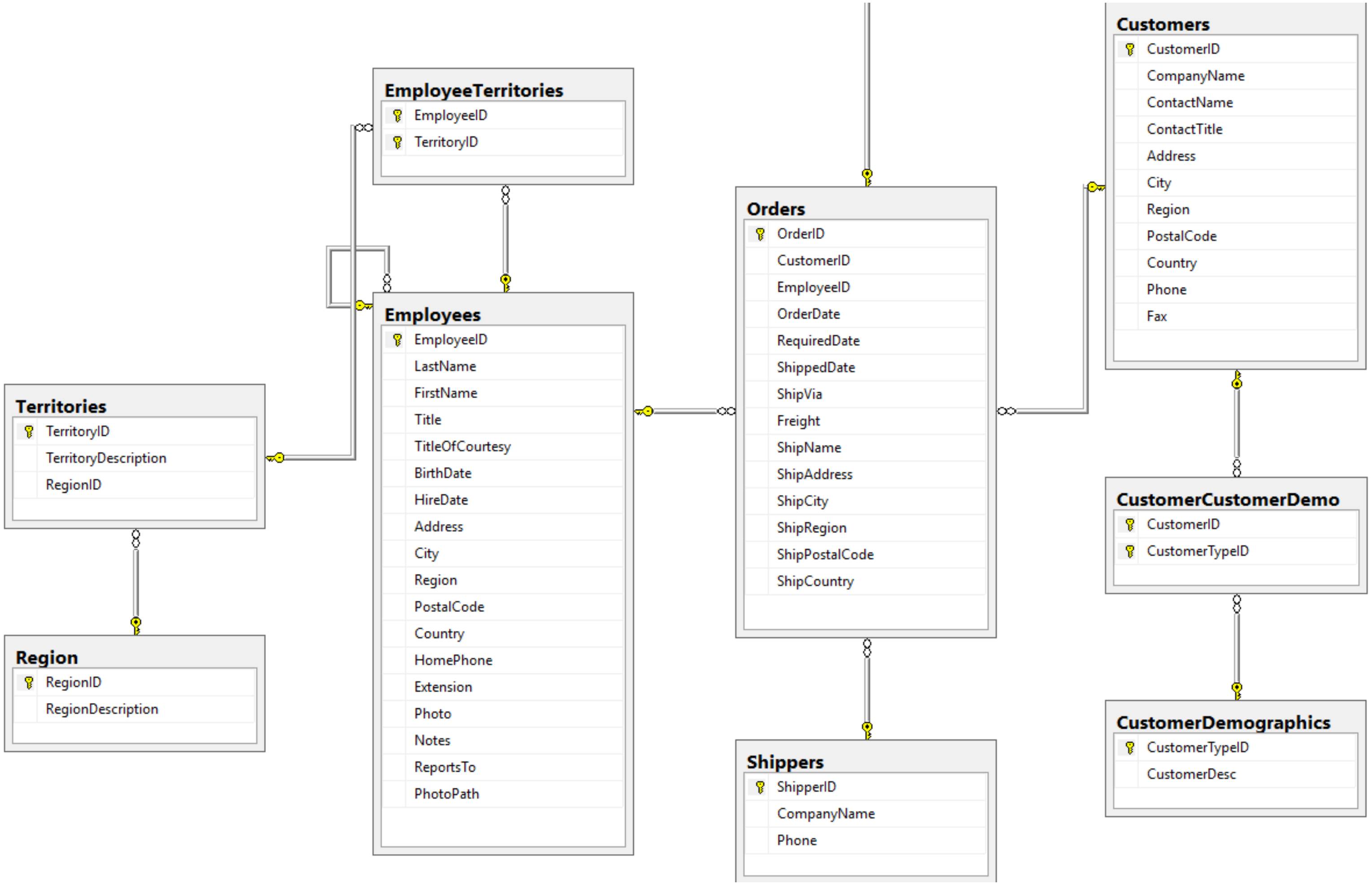
Lesson 1: Database Concepts

- Relational Databases
- Normalization
- Database Objects
- Data Types
- Levels of abstraction
- What is NULL
- Join operators

Relational Databases

- **Relational databases consist of many objects**
- **Most common object is a table**
  - Table is a logical structure for storing data
  - It is defined by its columns which represents the types of data to be stored
  - Every table should have a primary key
  - A table can also host foreign keys which describe its relation to another table
- **Tables are organized into schemas**

Example



EmployeeTerritories

- EmployeeID
- TerritoryID

Territories

- TerritoryID
- TerritoryDescription
- RegionID

Region

- RegionID
- RegionDescription

Employees

- EmployeeID
- LastName
- FirstName
- Title
- TitleOfCourtesy
- BirthDate
- HireDate
- Address
- City
- Region
- PostalCode
- Country
- HomePhone
- Extension
- Photo
- Notes
- ReportsTo
- PhotoPath

Orders

- OrderID
- CustomerID
- EmployeeID
- OrderDate
- RequiredDate
- ShippedDate
- ShipVia
- Freight
- ShipName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry

Shippers

- ShipperID
- CompanyName
- Phone

Customers

- CustomerID
- CompanyName
- ContactName
- ContactTitle
- Address
- City
- Region
- PostalCode
- Country
- Phone
- Fax

CustomerCustomerDemo

- CustomerID
- CustomerTypeID

CustomerDemographics

- CustomerTypeID
- CustomerDesc

Normalization

- **Database modeling - deciding which tables and columns, (together with other objects) are needed to support the application that consumes the data.**
- **Database normalization is the process that seeks to eliminate the need for multiple repetitions of the same data**
  - Division of large tables into smaller (and less redundant) tables and defining relationships between them
- **In June of 1970, Edgar F Codd introduced the concept of normalization known as First Normal Form (1NF)**
  - Codd also define the Second Normal Form (2NF) and Third Normal Form (3NF)

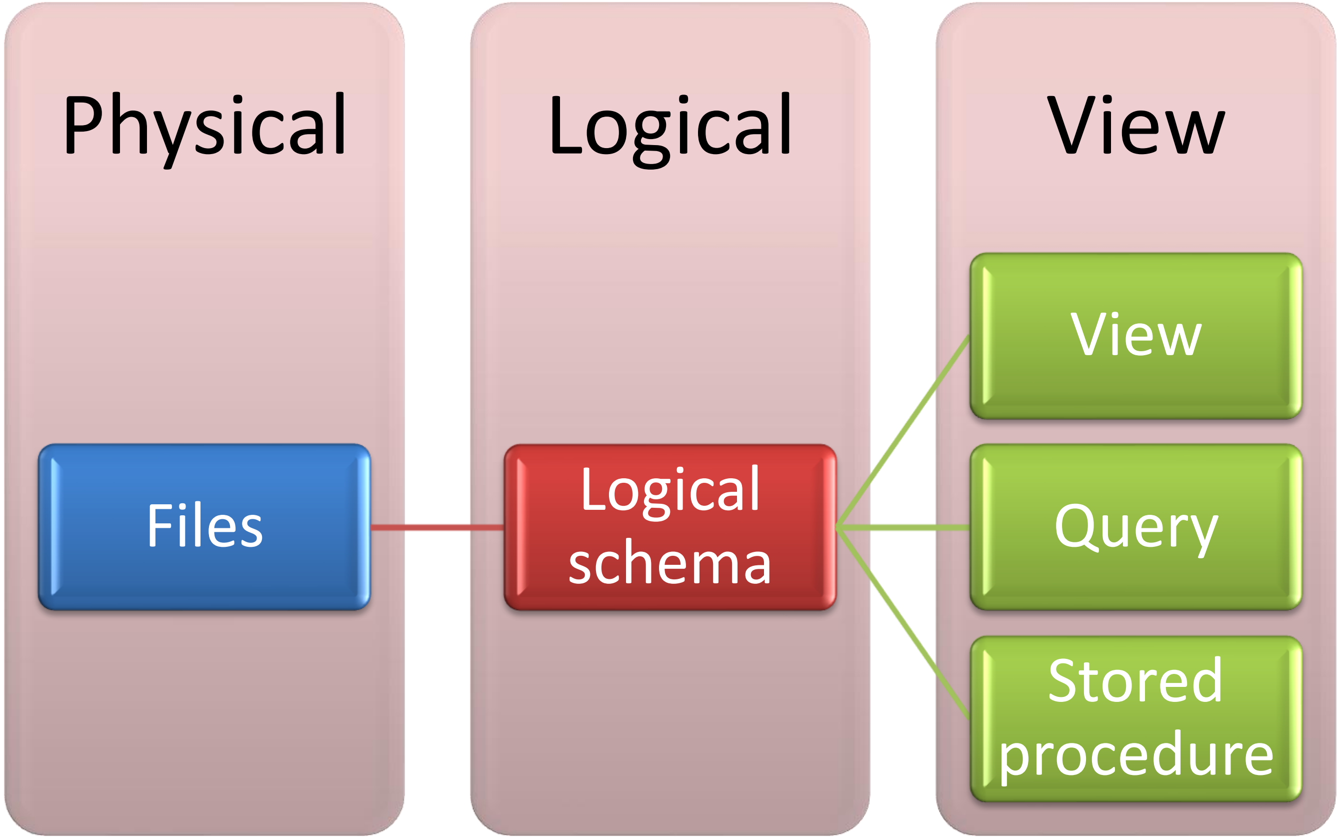
Database Objects

- **Stored procedures**
  - Objects for storing methods of action
- **Functions**
  - Objects that encapsulate formulas and calculations
- **Views**
  - Object that consists of one select-statement
- **Schemas**
  - An organizational object that can be described as a folder in a file system. All objects in a database must belong to a schema.
- **User-defined objects**
  - Objects that consist of an interpretation of a native SQL Server data type.

Data Types

- A data type defines how a value is structured, stored, and handled
- Structured data types are native SQL Server data types
  - int, char, varchar, datetime, binary, varbinary, money, decimal, geography, geometry, location, and so on.
- Semi-structured data types store its data in a structured manner internally and is usually handled by the database engine as large objects
  - xml
- Non-structured data types are used to store large amounts of data such as documents and binaries.
  - image, text, and ntext are usually called blob/clob

Levels of abstraction



What is NULL

- Definition of NULL
- Working with NULL Values
- NULL Value Practice by Example

Definition of NULL

- **NULL is unknown value**
- **NULL is not the same as zeros or blanks since they are both defined values**
- **Any arithmetic or comparison operation that involves NULL value will result with NULL value**
  - 50 + NULL → NULL
  - 50 < NULL → NULL
- **When one of the conditions in the WHERE clause results with NULL it is treated as FALSE**

Working with NULL Values

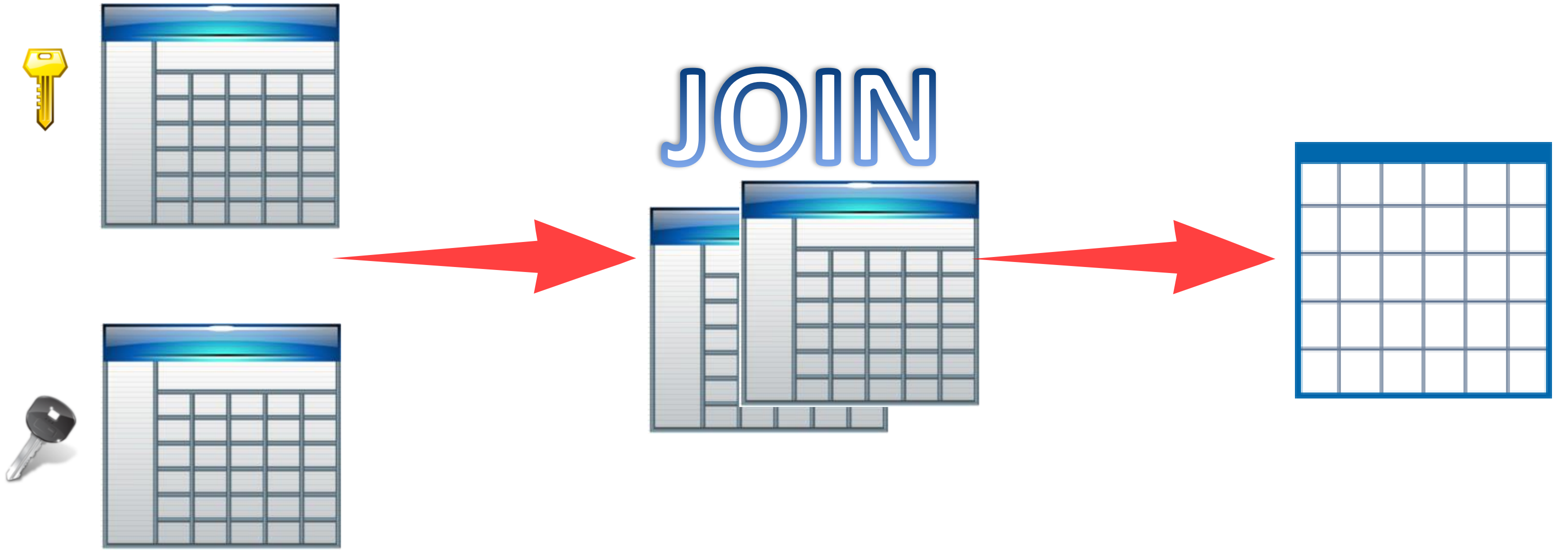
- NULL values can’t be compared using standard comparison operators
- Keywords IS NULL and IS NOT NULL are used to identify NULL values

```
SELECT Name, Color
FROM Production.Product
WHERE Color IS NULL
```

```
Name                                               Color
-------------------------------------------------- ---------------
Adjustable Race                                    NULL
Bearing Ball                                       NULL
BB Ball Bearing                                    NULL
....
LL Bottom Bracket                                  NULL
ML Bottom Bracket                                  NULL
HL Bottom Bracket                                  NULL
(248 row(s) affected)
```

What are Joins?

- Although not necessarily, related tables are created with common columns usually named primary and foreign key
- JOIN - combines data from two or more tables into a single result set (usually by using common columns)



Types of Joins

- **There are several types of JOIN operator :**
  - **INNER JOIN** – returns only matching rows
  - **OUTER JOIN** – returns the unmatched rows
  - **CROSS JOIN** – returns all rows from the LeftTable combined with all rows from the RightTable

```sql
SELECT LeftTable.Column1, RightTable.Column1
FROM LeftTable JoinType RightTable
ON JoinCondition
```

Querying using INNER JOIN

- Cross-section of two sets
- Resulting set will contain only those elements that are common to both tables

```
SELECT P.ProductNumber, R.Comments
FROM Production.Product P
INNER JOIN Production.ProductReview R
ON P.ProductID = R.ProductID
```

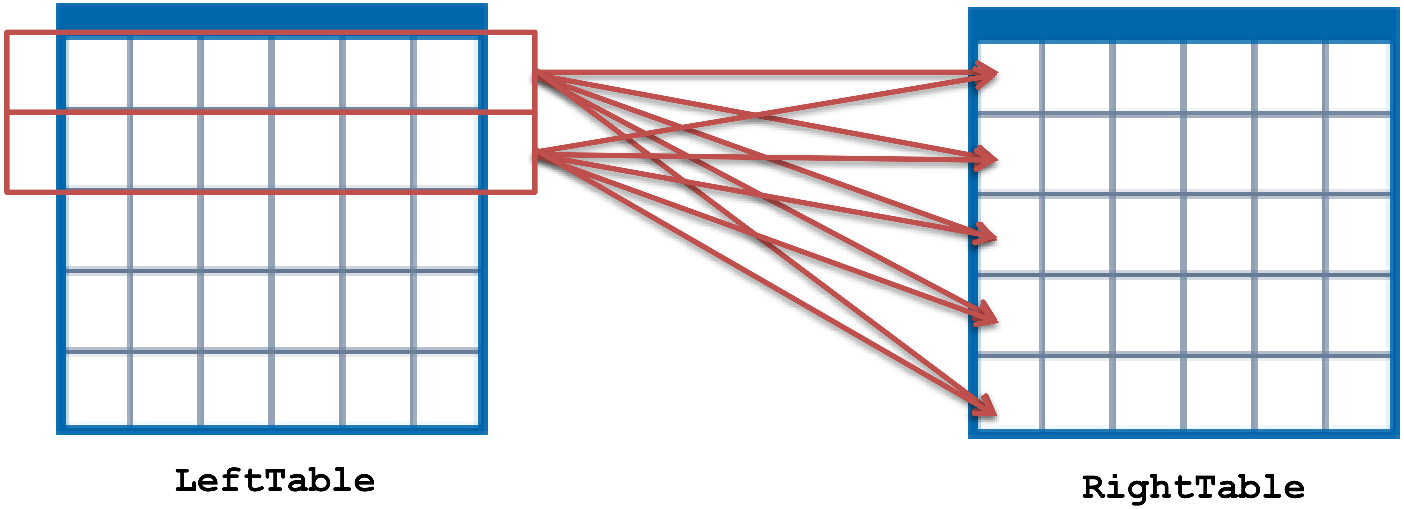
```
ProductNumber             Comments
------------------------- ----------------------------------------------
SO-B909-M                 I can't believe I'm singing the....
PD-M562                   A little on the heavy side, but overall ...
PD-M562                   Maybe it's just because I'm new to ....
BK-R64Y-40                The Road-550-W from Adventure Works Cycles ...

(4 row(s) affected)
```

Querying using OUTER JOIN

- **Returns even those data that does not have a match in the joining table**
- **There are three variations of OUTER JOIN operator:**
  - **LEFT OUTER JOIN –** returns all the rows from the LeftTable and only the matching rows from the RightTable
  - **RIGHT OUTER JOIN** – returns all the rows from the RightTable and only the matching rows from the LeftTable
  - **FULL OUTER JOIN** – returns all rows from both the LeftTable and the RightTable

Querying using CROSS JOIN

- Also known as Cartesian product
- Combines all rows from a LeftTable with all the rows in the RightTable



- CROSS JOIN should be treated with caution ⚠

Advanced Usage of Joins

- You will often require data from more than two tables
  - Multiple JOINs
- *Self-join* refers to any kind of join used to join a table to itself

```
SELECT  P.Name AS Product, L.Name AS Location, I.Quantity
FROM Production.Product P
       INNER JOIN Production.ProductInventory I
ON P.ProductID = I.ProductID
       INNER JOIN Production.Location L
ON L.LocationID = I.LocationID
```

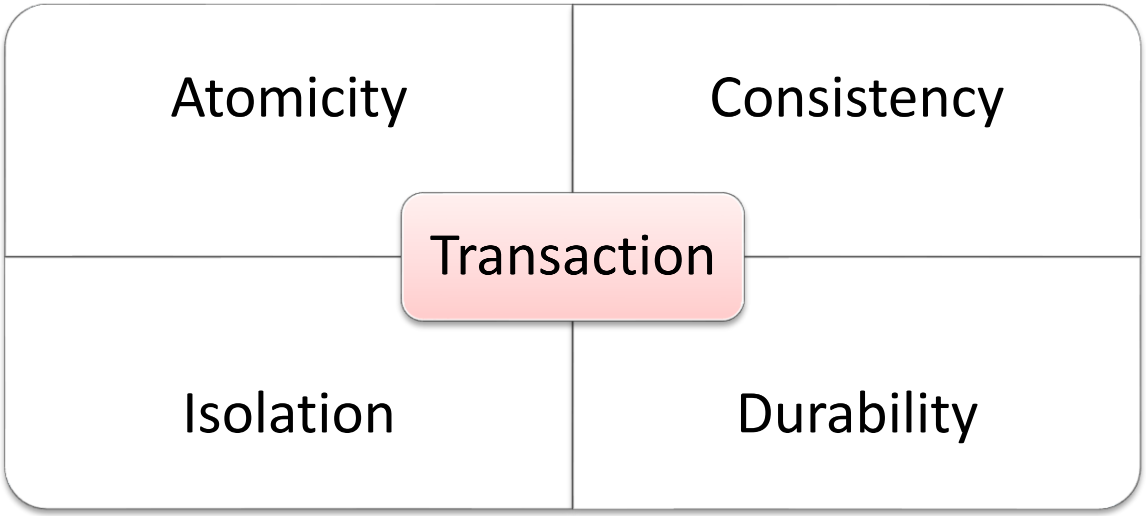
```
Product                              Location              Quantity
------------------------------------ ---------------------- ----------------------
Adjustable Race                      Tool Crib              408
Adjustable Race                      Miscellaneous Storage  324
....
Road-750 Black, 52                   Final Assembly         116

(1069 row(s) affected)
```

Lesson 2: Fundamentals About Transactions

- Transaction Fundamentals
- Transactions and the Database Engine
- Basic Transaction Statement Definitions
- Using Nested Transactions

Transaction Fundamentals

- **Purpose of transaction is to keep data consistency.**
  - Question is how?
- **Answer is ACID.**
- **When transaction starts it needs to satisfied following rules.**



Atomicity

Consistency

Transaction

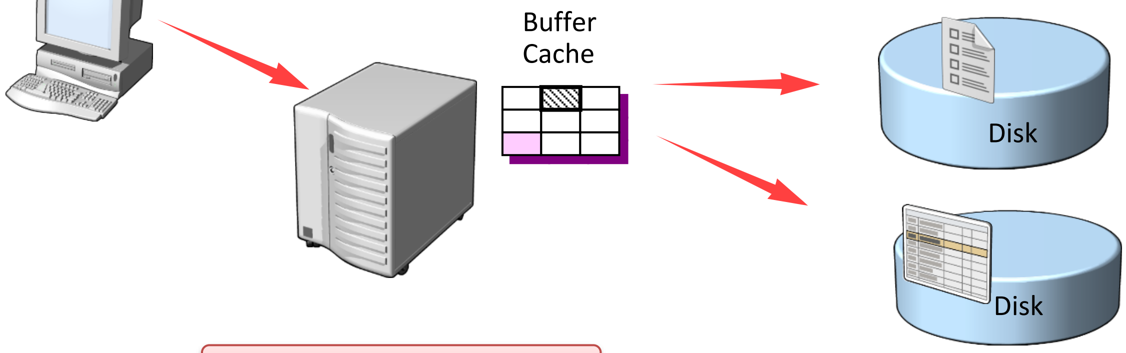
Isolation

Durability

- **If any of those four rules is fail, transaction needs to be rollback**

Transactions and the Database Engine

- **Database engine provides mechanism to handle transaction on proper way.**
  - WAL – write ahead log is key components
  - Locking - Isolation levels of transactions
  - Logging - ensure transaction durability



1 Data modification is sent by application

2 Data pages are located in, or read into, buffer cache and modified

3 Modification is recorded in transaction log on disk

4 Checkpoint writes committed transactions to database

Basic Transaction Statement Definitions

- **BEGIN TRANSACTION – is appoint in TSQL code from where all parts are treated as one for keeping data consistent.**

- **COMMIT TRANSACTION - all data modifications performed since the start of the transaction are permanent**

- **ROLLBACK TRANSACTION - Rolls back an explicit or implicit transaction to the beginning of the transaction**

  - ***You cannot rollback transaction after COMMIT***

Questions