B2L2

Block 2 Lektion 2

Relational Databases

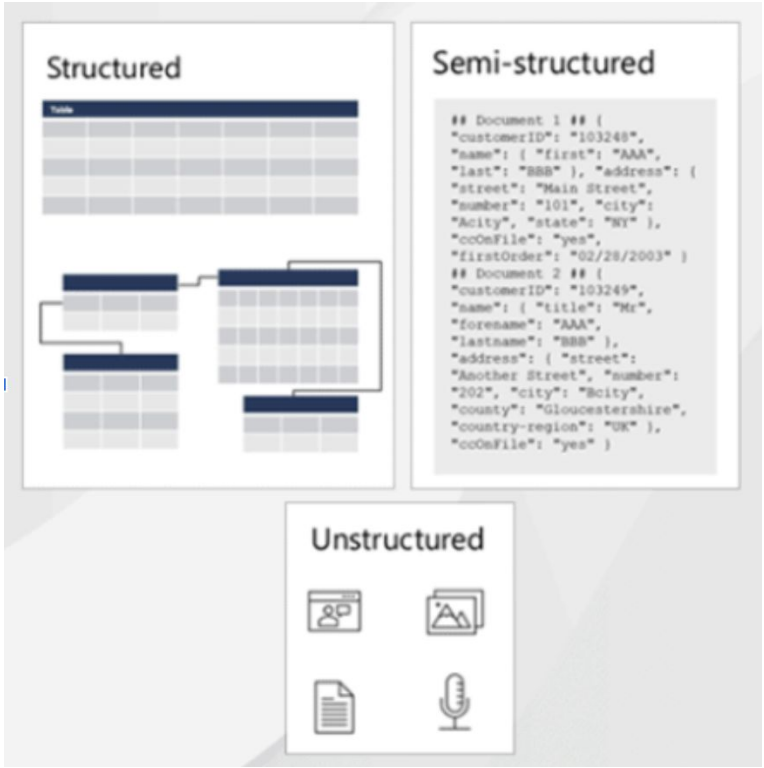
And how they differ from NoSQL databases

Topics in this Module

- Structured, semi-structured, unstructured data
- Vertical (scaling up) & Horizontal scaling (scaling out)
- Relational (SQL) vs Non-relational data (NoSQL)
- Normalisation
- SQL Joins
- SSMS Database diagrams (ERD)

Structured, Semi-structured and Unstructured Data

- Structured Data
  - Tables (rows & columns)
  - Cardinal relationships (data in one table relates to data in another table)
- Semi-structured
  - .json
  - .xml
  - .csv
- Unstructured
  - Machine generated/encoded e.g:
    - .png
    - .mp3
    - .mp4
    - .pdf

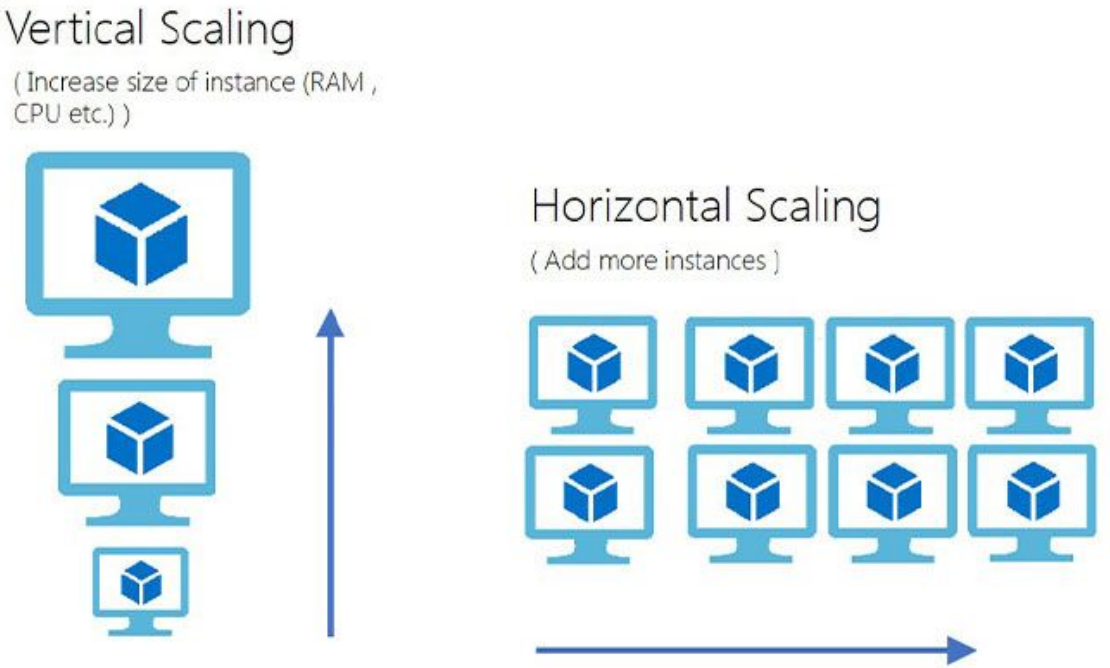


Vertical Scaling - SQL databases scale up well

Throughput: Typically measured in transactions per second (TPS) or transactions per minute (TPM). measures the overall performance a system. Throughput can be a bottleneck when in comes to database CRUD scenarios.

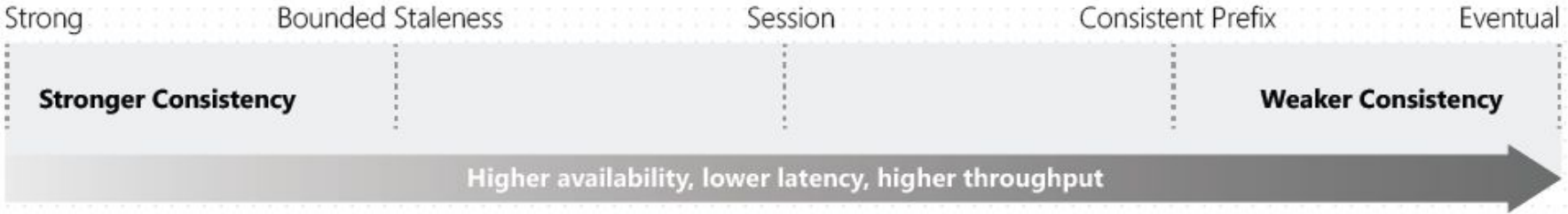
Throughput depends on the following factors:

- The specifications of the host computer
- The processing overhead in the software
- The layout of data on disk
- The degree of parallelism that both hardware and software support
- The types of transactions being processed



Horizontal Scaling (database replication) - Not ideal for an SQL DB

- Database Throughput bottleneck addressed by
  - **Partitioning/sharding** across multiple machines (Each partition holds different tables)
    - Consistency is maintained at the cost of increased read latency
    - Increased capacity for write operations
    - Read operations such as *joins* become expensive as multiple machines are involved.
    - Risk: operation failure: deadlock (If multiple updates for a row do not come from the same connection - each partition has its own connection - it would cause blocking or deadlocks)- (roll-back with stored procedures)
  - **Replication**
    - entire db is replicated (redundancy = increased fault tolerance) at the cost of consistency (unless we enforce higher consistency level)
    - Write operations become expensive as we must write to every database instance.



SQL vs NoSQL (Relational vs Non-relational)

NoSQL: availability over consistency

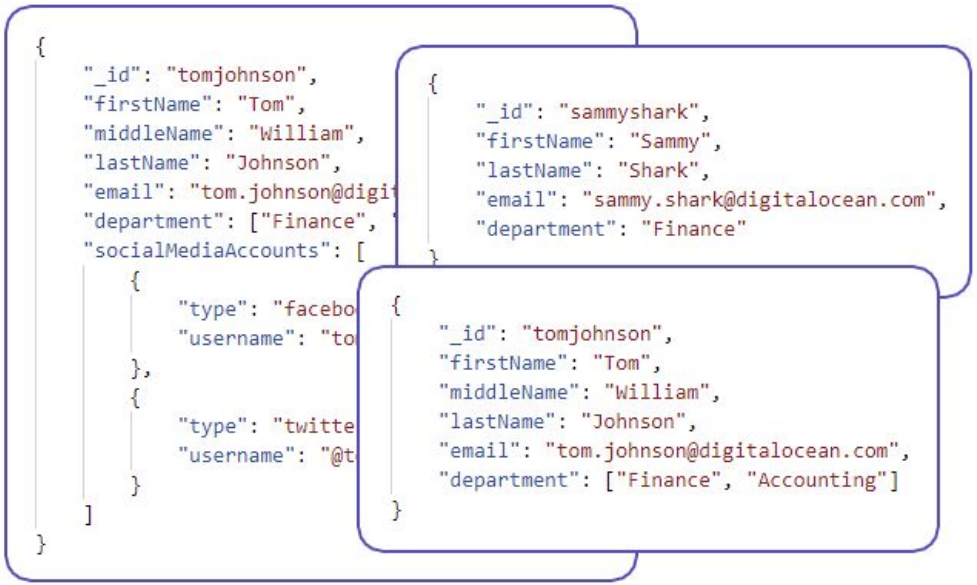
- Large data quantities, faster write/read, document schema may change over time
- Non-relational and schemaless, more flexible but increases application responsibilities (e.g. handle “joins” & consistency scenarios).
- easier to scale horizontally, allowing for increased reliability (availability & resiliency), frequent changes,

NoSQL: BSE (BASE)

- **B**asic **A**vailability
  - Low consistency allows for high availability
- Soft State
  - Doesn’t have to be write consistent, may serve “older state”
- Eventual Consistency
  - Availability over consistency (at some point it will likely be consistent)

NoSQL - complex unstructured data: document, keyvalue pairs, graph, cache (data in memory) etc for large dataquantities

NoSQL document (json object) example - notice the variation between documents



Structured data

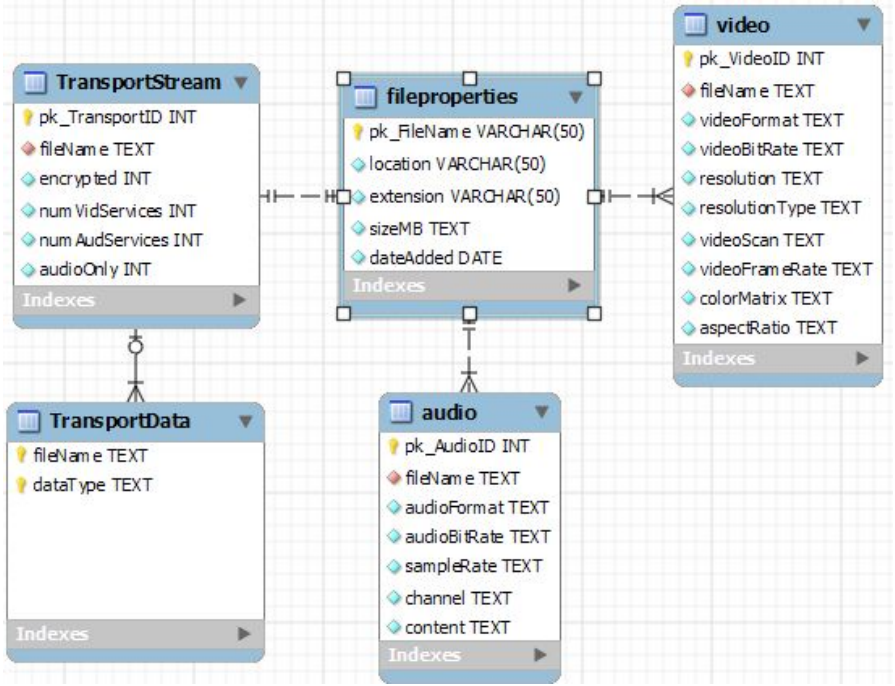
SQL: Consistency over availability

- Banking & online transactions. Registering customers, employees, software licenses etc.

SQL: ACID

- **A**tomicity:
  A field holds only one value. *Transaction must complete or revert to previous state (money never lost) - success else rollback*
- **C**onsistency:
  Constraints to enforce data states (only allowed values: not null, length etc)
- **I**solation:
  Concurrent execution safe (ensures sequential transaction outcome)
- **D**urability
  Data is persisted in non-volatile (safe from crash/power loss) memory e.g. SSD/HDD (NVS Devices).

Example: Structured way of storing metadata about Blob (stored in object storage)



Normalisation - reduces application responsibilities

- Normalisation mitigates data *redundancy* and *inconsistent dependencies* and facilitates maintaining *data integrity (see image) e.g. a person cannot have two dates of birth*.
- Each step in the mitigation process entails applying a rule (from 1st - 6th Normal form)
- "*Fully normalised*" data is easy to work with, easy to interpret, easy to control access etc!
- Consider the following example (non-normalised data) in the table below:

Employee

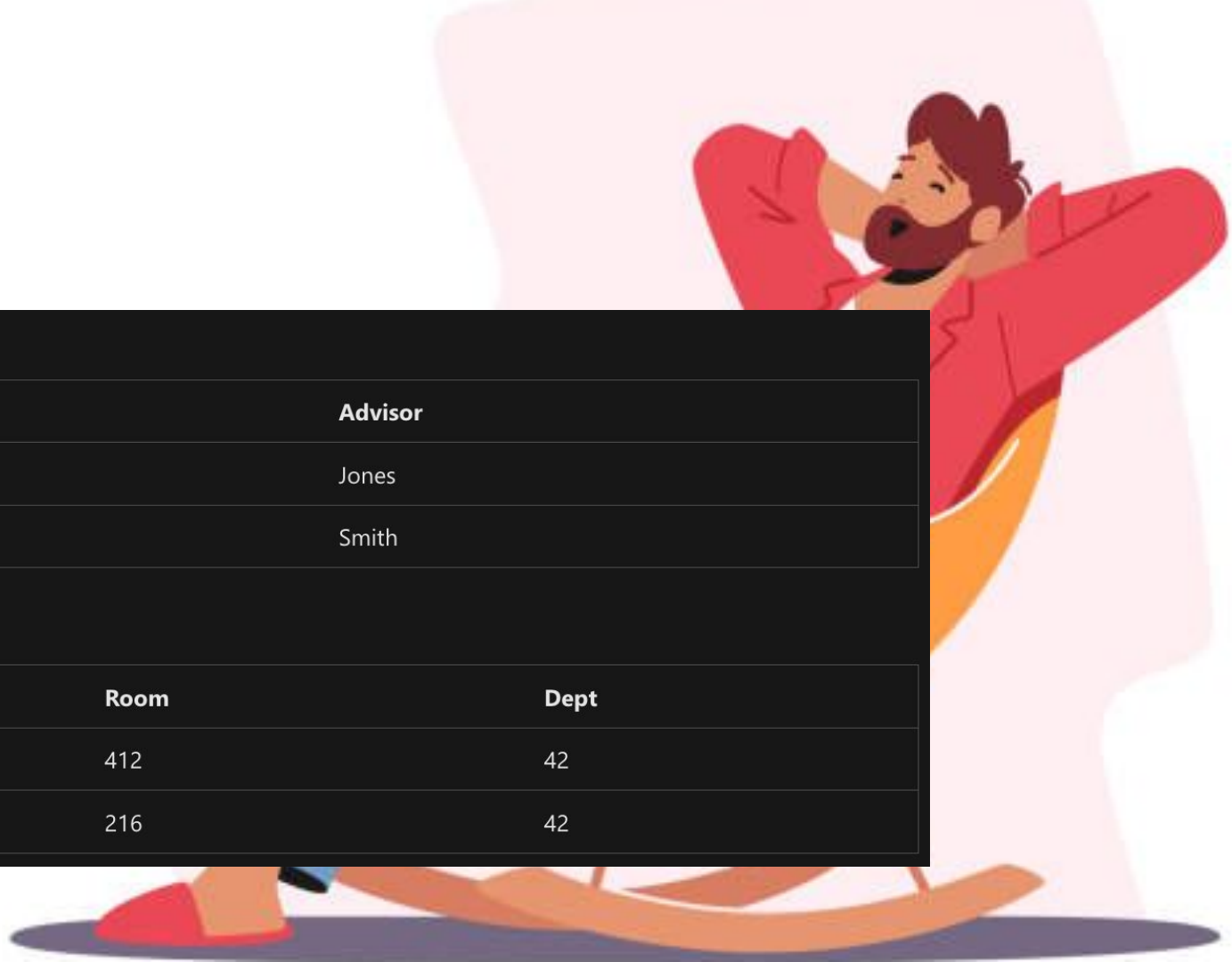
Man#	Name	Birthdate	JobHistory			Children	
123	Michael Swart	Nov. 22	**JobDate**	**Title**	**SalaryHistory**	**Childname**	**Birthyear**
			2000	Lackey	**SalaryDate** / **Salary**: 2000 / $1,000,000; 2001 / $2,000,000	Mini-me 1	2000
			2002	Senior Lackey	**SalaryDate** / **Salary**: 2002 / $3,000,000; 2003 / $4,000,000	Mini-me 2	2002

Probably the only time you'll see data structured this way :-). 3NF is considered standard and fully normalised in most scenarios

From this…

Student#	Advisor	Adv-Room	Class1	Class2	Class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	101-07	143-01	179-04

To this…



Students:

Student#	Advisor
1022	Jones
4123	Smith

Faculty:

Name	Room	Dept
Jones	412	42
Smith	216	42

Code Along

Let’s explore the concept of

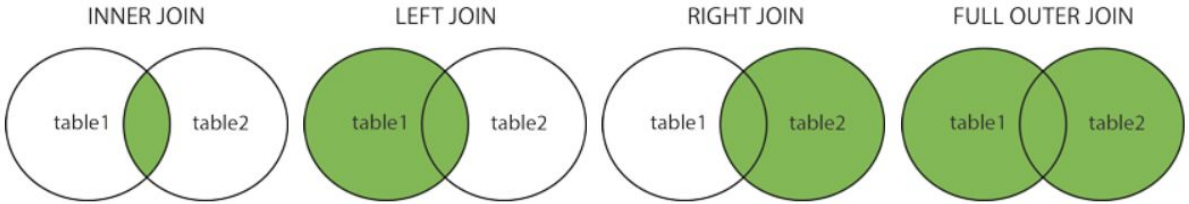
Normalisation together!

JOINS

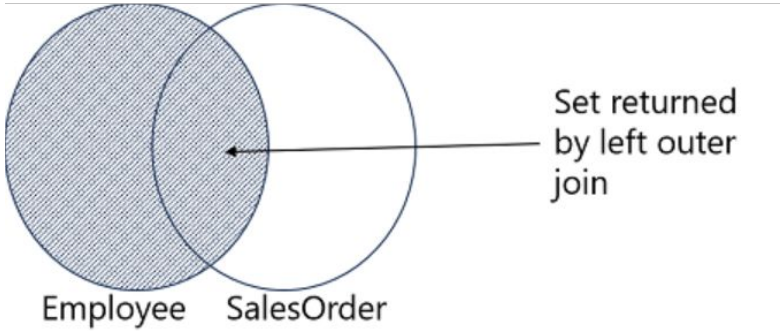
Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

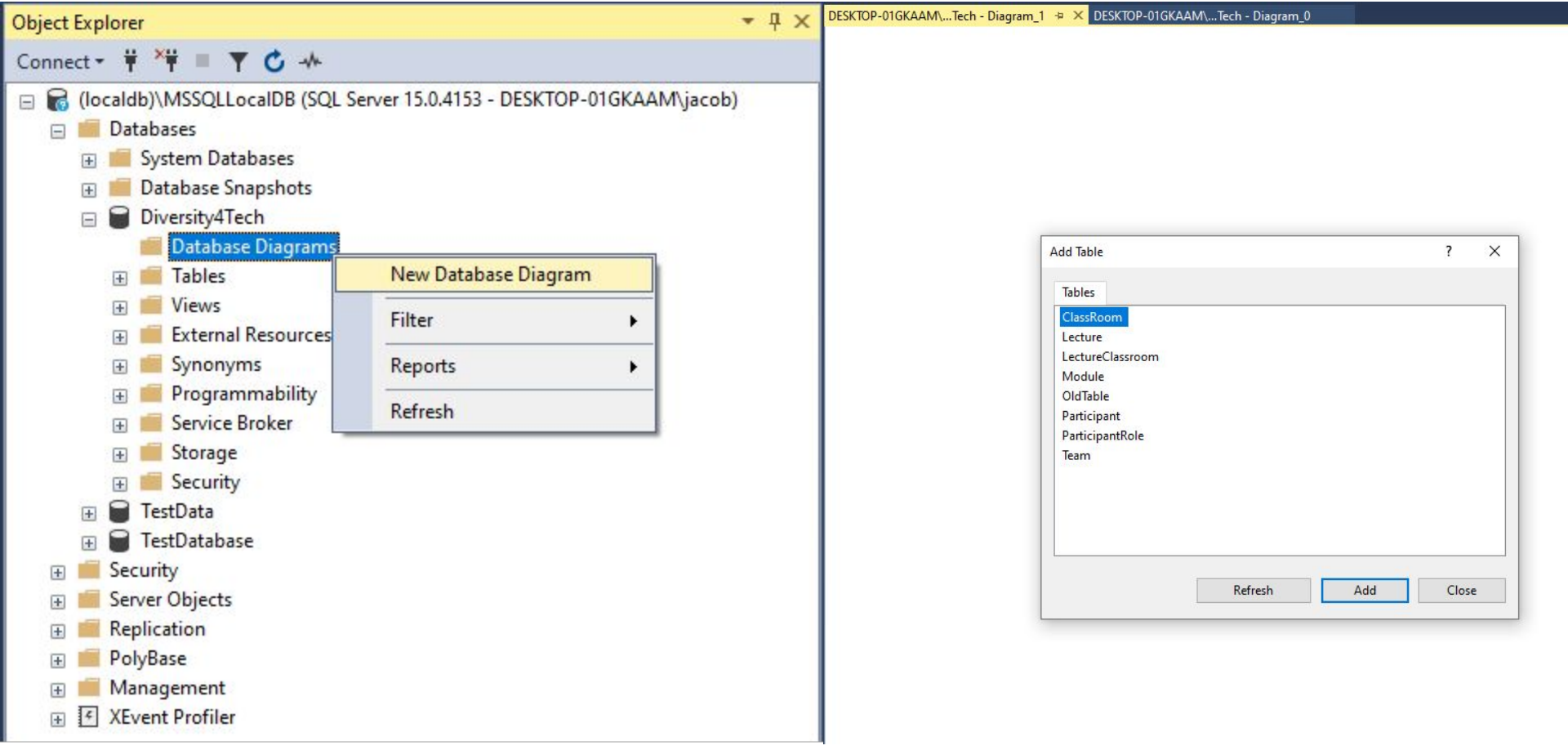
- `(INNER) JOIN` : Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN` : Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN` : Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN` : Returns all records when there is a match in either left or right table

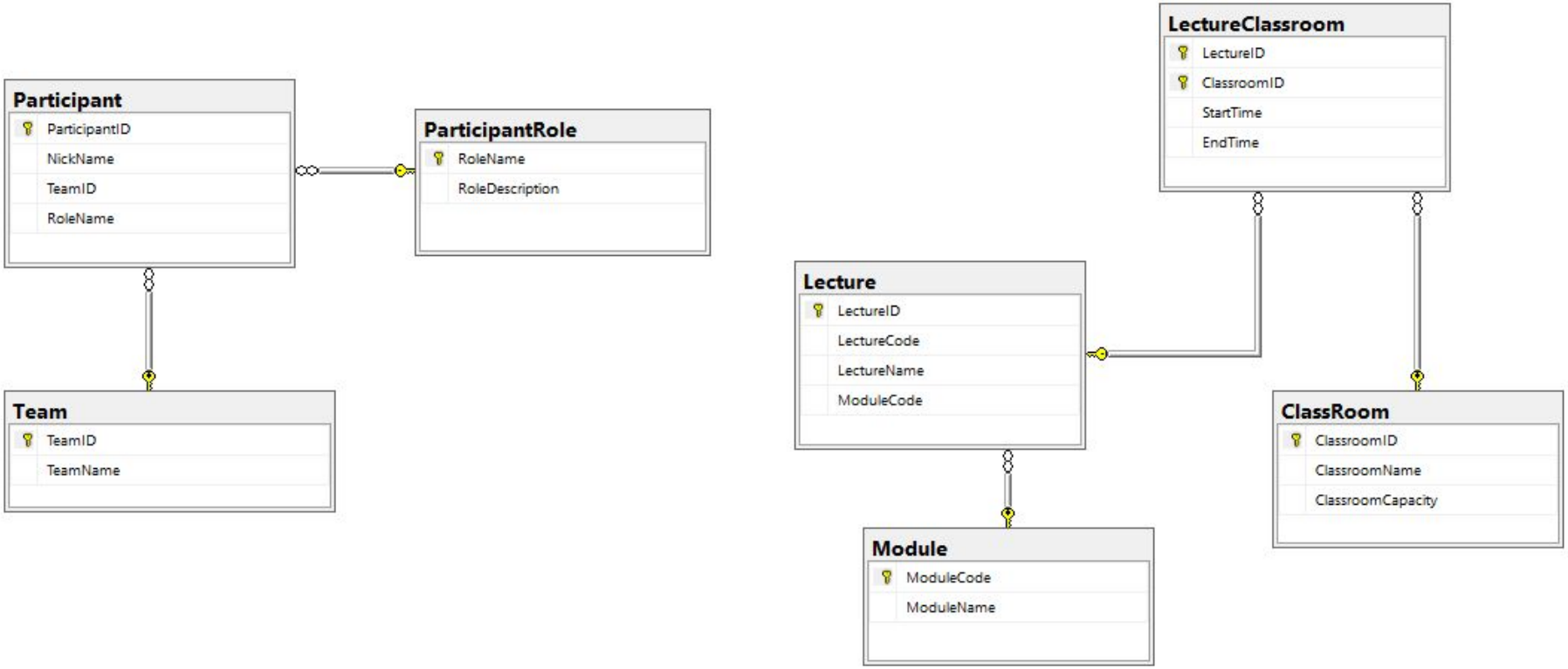



```sql
SELECT emp.FirstName, ord.Amount
FROM HR.Employee AS emp
LEFT OUTER JOIN Sales.SalesOrder AS ord
    ON emp.EmployeeID = ord.EmployeeID;
```



SSMS Database Diagrams
(Entity Relationship Diagrams)


Object Explorer
Connect
(localdb)\MSSQLLocalDB (SQL Server 15.0.4153 - DESKTOP-01GKAAM\jacob)
Databases
System Databases
Database Snapshots
Diversity4Tech
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
TestData
TestDatabase
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
New Database Diagram
Filter
Reports
Refresh
DESKTOP-01GKAAM\...Tech - Diagram_1
DESKTOP-01GKAAM\...Tech - Diagram_0
Add Table
Tables
ClassRoom
Lecture
LectureClassroom
Module
OldTable
Participant
ParticipantRole
Team
Refresh
Add
Close


Participant
ParticipantID
NickName
TeamID
RoleName
ParticipantRole
RoleName
RoleDescription
Team
TeamID
TeamName
LectureClassroom
LectureID
ClassroomID
StartTime
EndTime
Lecture
LectureID
LectureCode
LectureName
ModuleCode
ClassRoom
ClassroomID
ClassroomName
ClassroomCapacity
Module
ModuleCode
ModuleName

Further Reading

- Structured vs. Unstructured Data: What’s the Difference? | IBM
- Relational vs. Non-Relational Databases
- SQL vs. NoSQL: What's the difference?
- Database normalization description - Office | Microsoft Learn
- Database Normalization – Normal Forms 1nf 2nf 3nf Table Examples
- Primary and Foreign Key Constraints - SQL Server | Microsoft Learn
- Create Foreign Key Relationships - SQL Server | Microsoft Learn