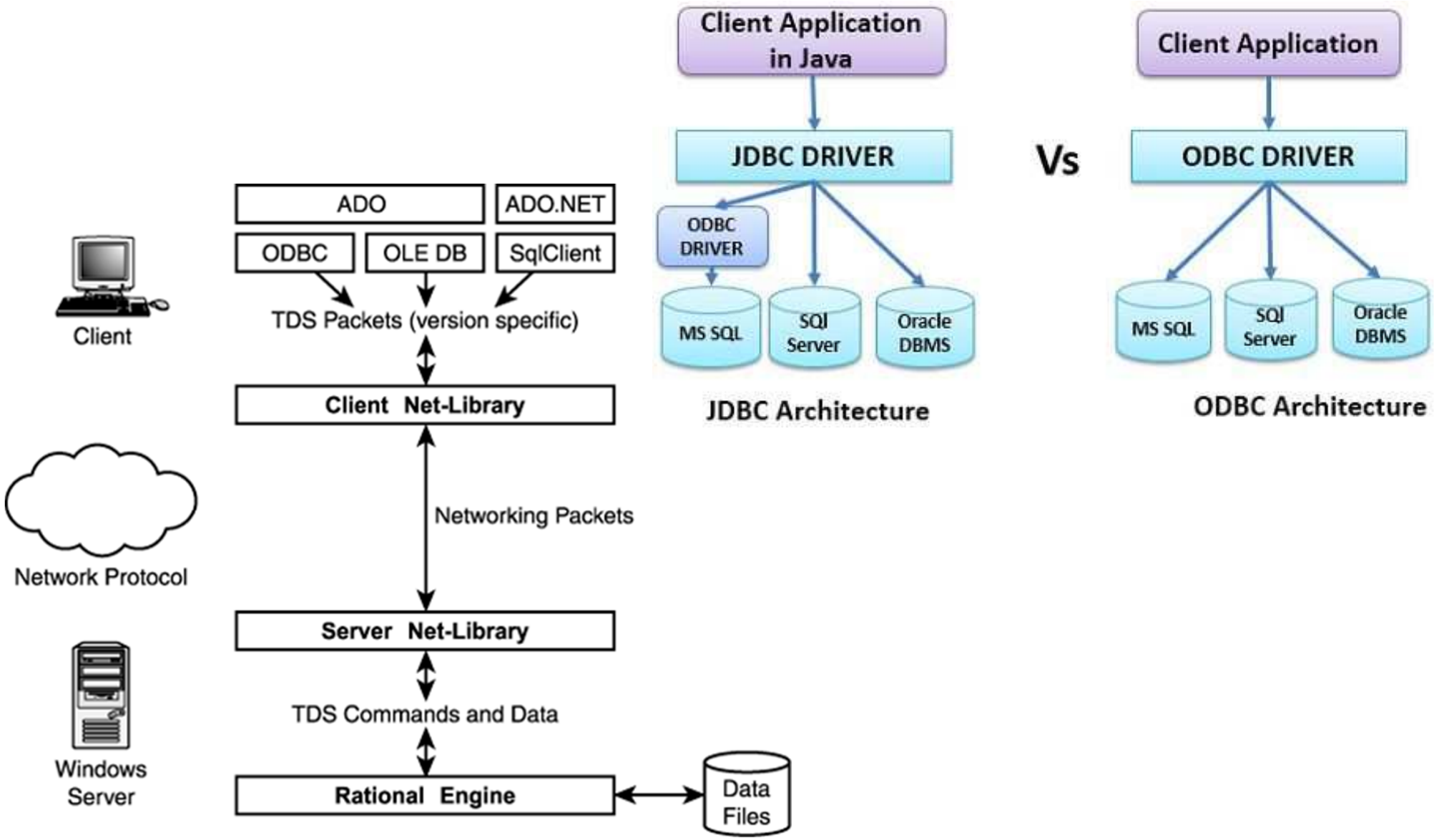
Database Systems

Application Development and Other DBMSs

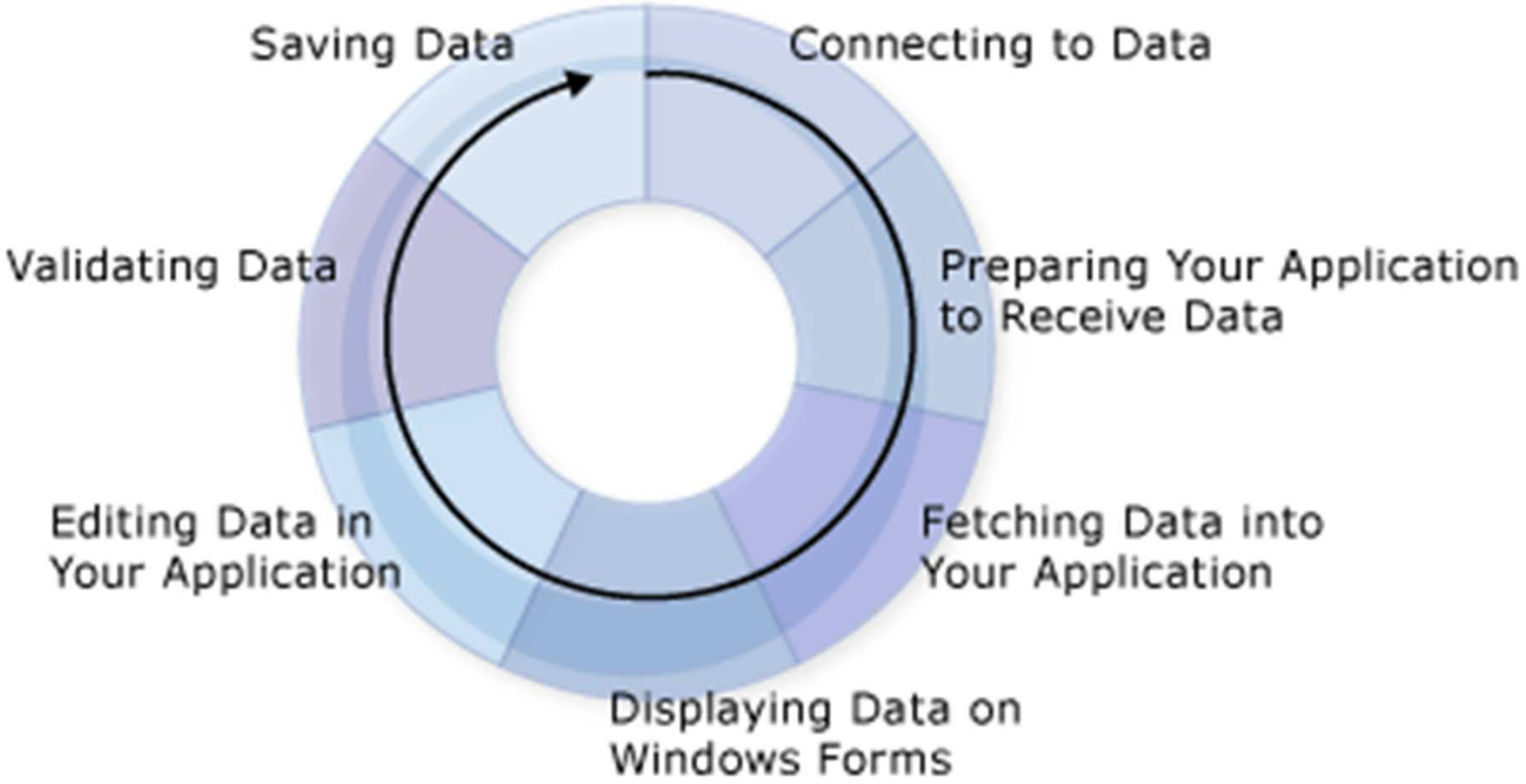
Application Development:

ADO.NET, Entity Framework
Language Integrated Query (LINQ)
Python and ORM

General Architecture and Drivers



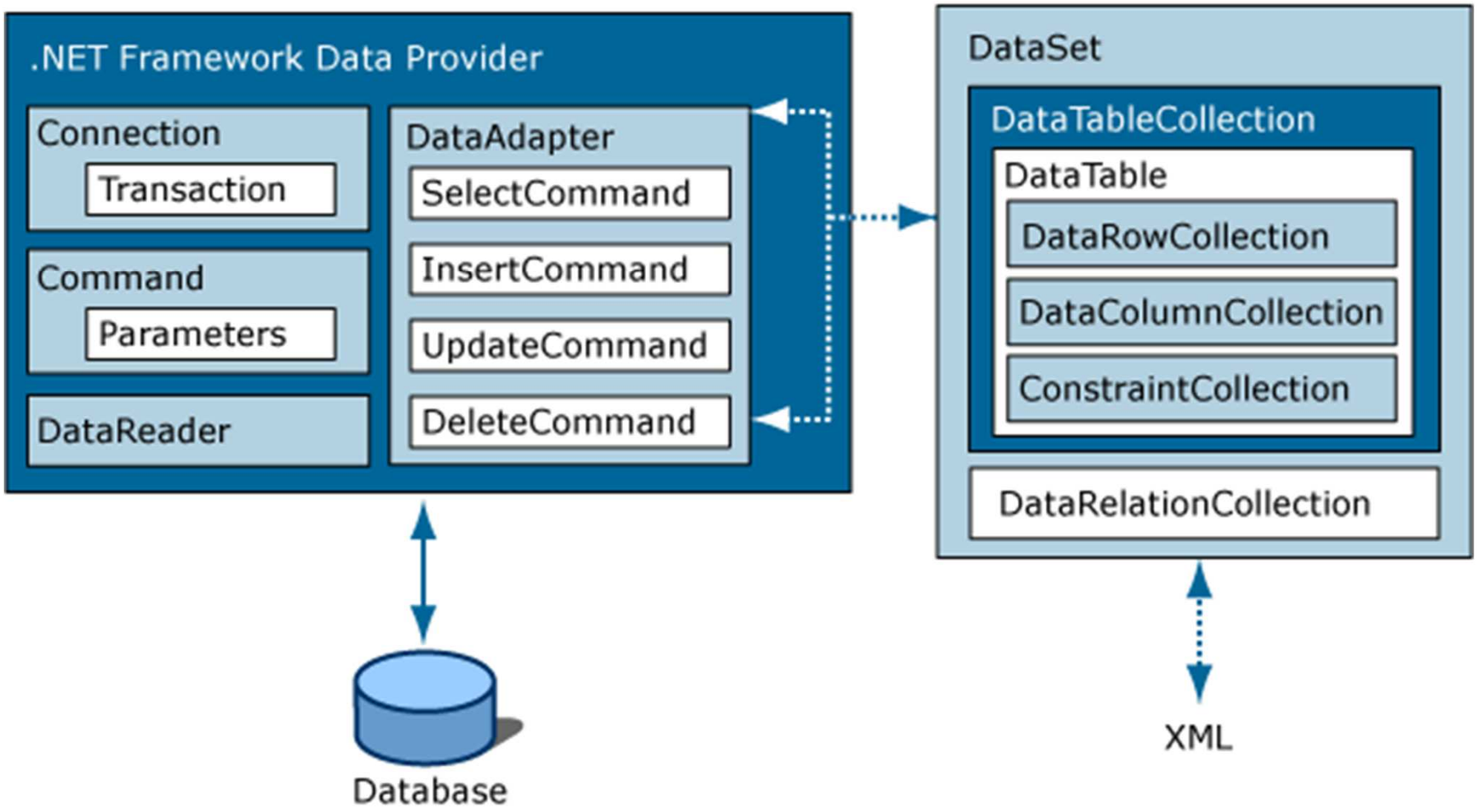
Creating Data Applications



http://msdn.microsoft.com/en-us/library/h0y4a0f6(v=vs.110).aspx

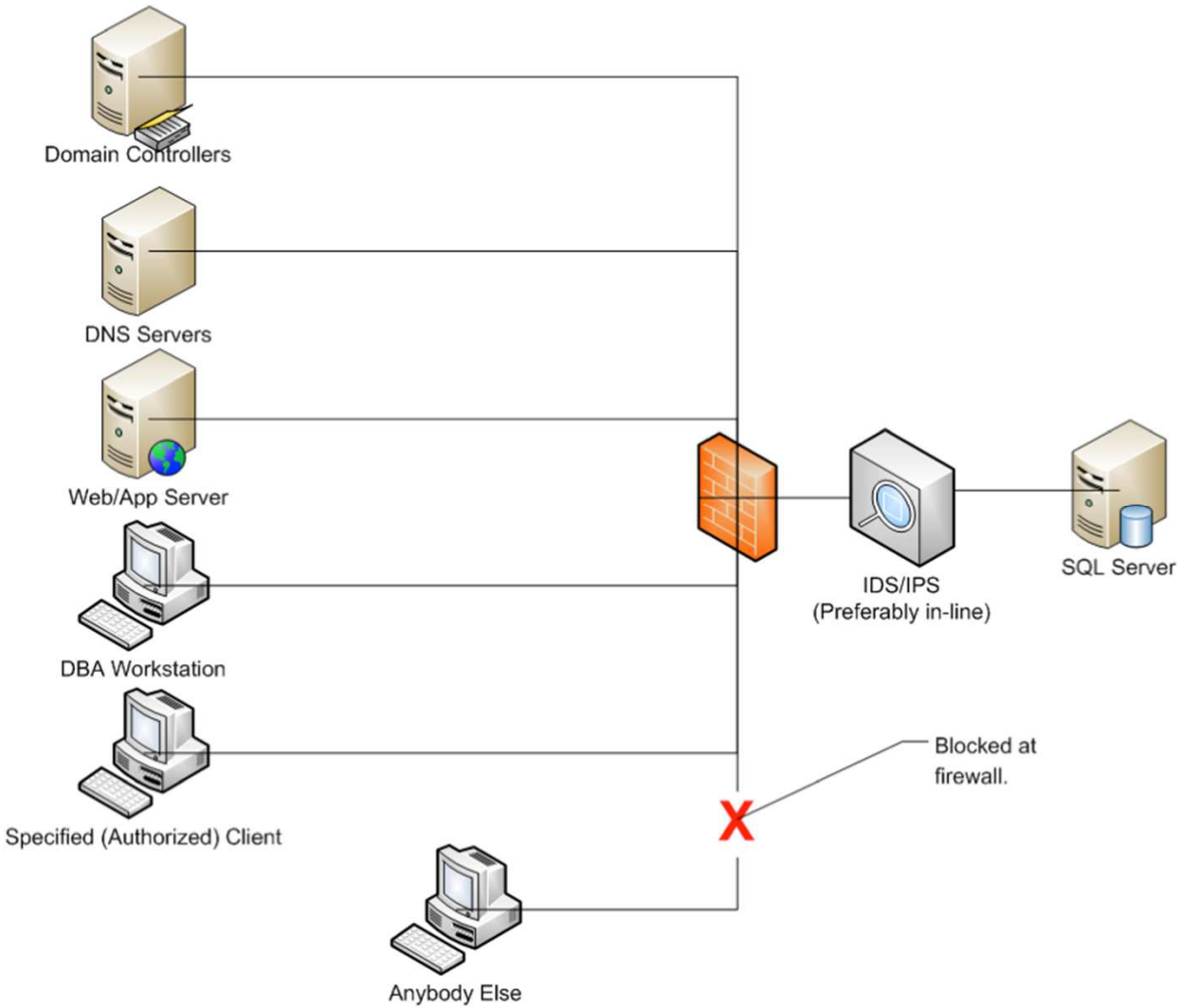
- Instead of "Windows Forms" you can of course create *web* or *mobile* applications
  - Other five steps are similar

ADO.NET Architecture



Read full article here:
http://msdn.microsoft.com/en-us/library/vstudio/27y4ybxw(v=vs.100).aspx

Database server in a network



ADO.NET

- A data access technology from the .NET Framework
- Provides communication between relational and non-relational systems through a common set of components
- A part of the base class library that is included with the .NET Framework.
- Mostly for relational
  - Supports non-relational

ADO.NET

- API designed for Visual Basic .NET and C#, providing database access facilities similar to JDBC/ODBC
  - Partial example of ADO.NET code in C#

```
using System.Data;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString =
            "Data Source=(local);Initial Catalog=Northwind;"
            + "Integrated Security=true";

        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from dbo.products "
                + "WHERE UnitPrice > @pricePoint "
                + "ORDER BY UnitPrice DESC;";
```

http://msdn.microsoft.com/en-us/library/dw70f090

ADO.NET DataReader

```
static void HasRows(SqlConnection connection)
{
    using (connection)
    {
        SqlCommand command = new SqlCommand(
          "SELECT CategoryID, CategoryName FROM Categories;",
          connection);
        connection.Open();

        SqlDataReader reader = command.ExecuteReader();

        if (reader.HasRows)
        {
            while (reader.Read())
            {
                Console.WriteLine("{0}\t{1}", reader.GetInt32(0),
                    reader.GetString(1));
            }
        }
        else
        {
            Console.WriteLine("No rows found.");
        }
        reader.Close();
    }
}
```

Results from Multiple Readers

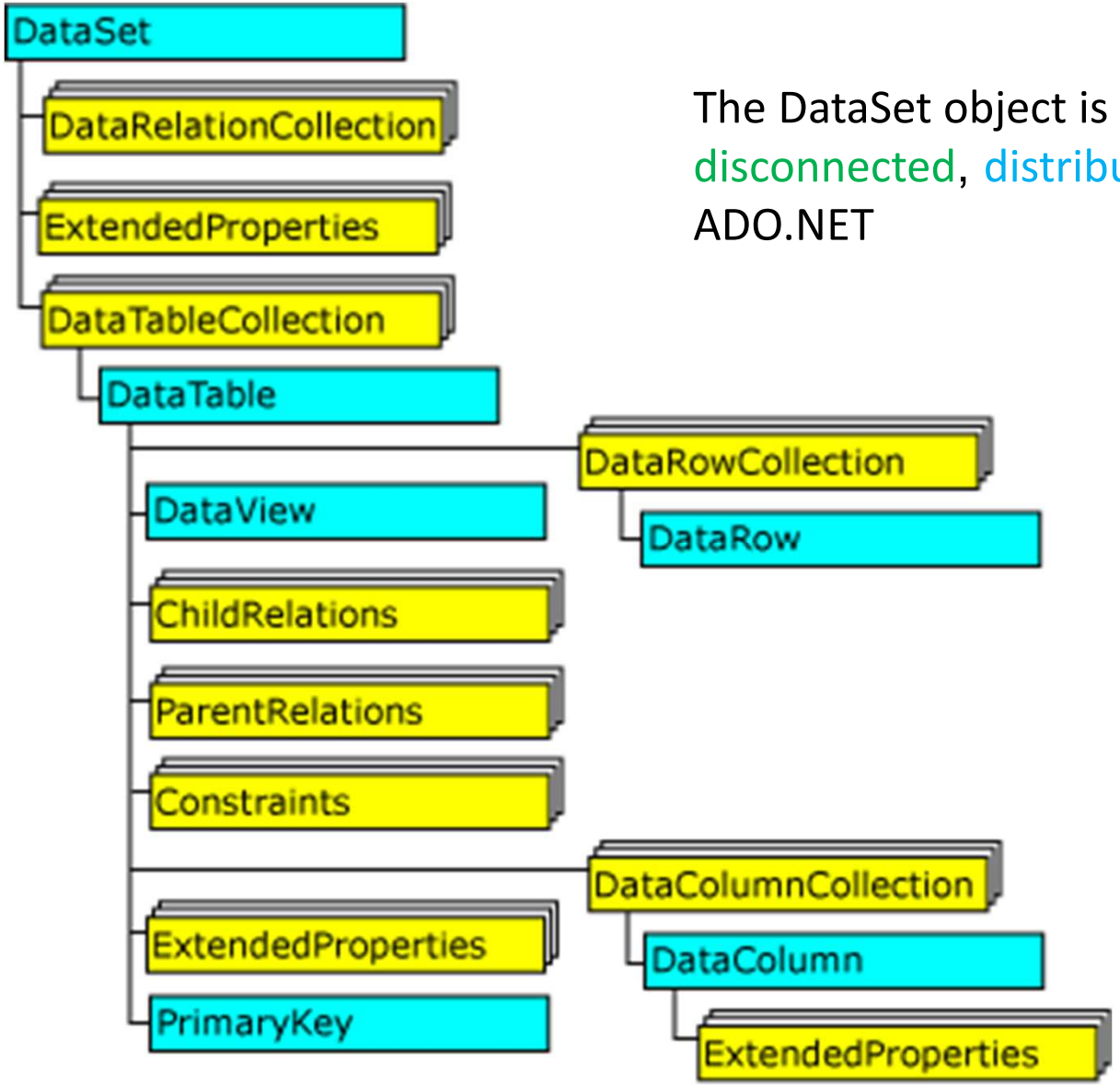
```
static void RetrieveMultipleResults(SqlConnection connection)
{
    using (connection)
    {
        SqlCommand command = new SqlCommand(
          "SELECT CategoryID, CategoryName FROM dbo.Categories;" +
          "SELECT EmployeeID, LastName FROM dbo.Employees",
          connection);
        connection.Open();

        SqlDataReader reader = command.ExecuteReader();

        while (reader.HasRows)
        {
            Console.WriteLine("\t{0}\t{1}", reader.GetName(0),
                reader.GetName(1));

            while (reader.Read())
            {
                Console.WriteLine("\t{0}\t{1}", reader.GetInt32(0),
                    reader.GetString(1));
            }
            reader.NextResult();
        }
    }
}
```

The DataSet object model



The DataSet object is central to supporting disconnected, distributed data scenarios with ADO.NET

Relations in DataSet

```
DataRelation customerOrdersRelation =
    customerOrders.Relations.Add("CustOrders",
    customerOrders.Tables["Customers"].Columns["CustomerID"],
    customerOrders.Tables["Orders"].Columns["CustomerID"]);

foreach (DataRow custRow in customerOrders.Tables["Customers"].Rows)
{
    Console.WriteLine(custRow["CustomerID"].ToString());

    foreach (DataRow orderRow in custRow.GetChildRows(customerOrdersRelation))
    {
        Console.WriteLine(orderRow["OrderID"].ToString());
    }
}
```

Entity Framework for ADO.NET

- An open source object-relational mapping (ORM) framework for ADO.NET, part of .NET Framework
- A set of technologies in ADO.NET that support the development of data-oriented software applications
- Designed to enable developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema.
- Enables developers to work with data in the form of domain-specific objects and properties
  - E.g. customers and customer addresses
  - No concern with the underlying database tables and columns

Entity Framework for ADO.NET – cont.

- Goal : decrease the amount of code and maintenance required for data-oriented applications
- Developers can work at a higher level of abstraction
- Other solutions for O/R mapping:
  - List on next slide
  - Write it yourself! (not recommended)

List of ORM software (some)

PHP [edit]

- CakePHP, ORM and framework for PHP 5, open source (scalars, arrays, objects); based on database introspection, no class extending
- CodeIgniter, framework that includes an ActiveRecord implementation
- Doctrine, open source ORM for PHP 5.2.3, 5.3.X. Free software (MIT)
- FuelPHP, ORM and framework for PHP 5.3, released under the MIT license. Based on the ActiveRecord pattern.
- Laravel, framework that contains an ORM called "Eloquent" an ActiveRecord implementation.
- Propel, ORM and query-toolkit for PHP 5, inspired by Apache Torque, free software, MIT
- Qcodo, ORM and framework for PHP 5, open source
- QCubed, A community driven fork of Qcodo
- Redbean, ORM layer for PHP 5, creates and maintains tables on the fly, open source, BSD
- Skipper, visualization tool and a code/schema generator for PHP ORM frameworks, commercial
- Yii, ORM and framework for PHP 5, released under the BSD license. Based on the ActiveRecord pattern.
- Zend Framework, framework that includes a table data gateway and row data gateway implementations.

Python [edit]

- Django, ORM included in Django framework, open source
- SQLAlchemy, open source
- SQLObject, open source
- Storm, open source (LGPL 2.1) developed at Canonical Ltd.
- Tryton, open source
- web2py, the facilities of an ORM are handled by the DAL in web2py, open source
- Odoo - Formerly known as OpenERP, It is an Open Source ERP in which ORM is included

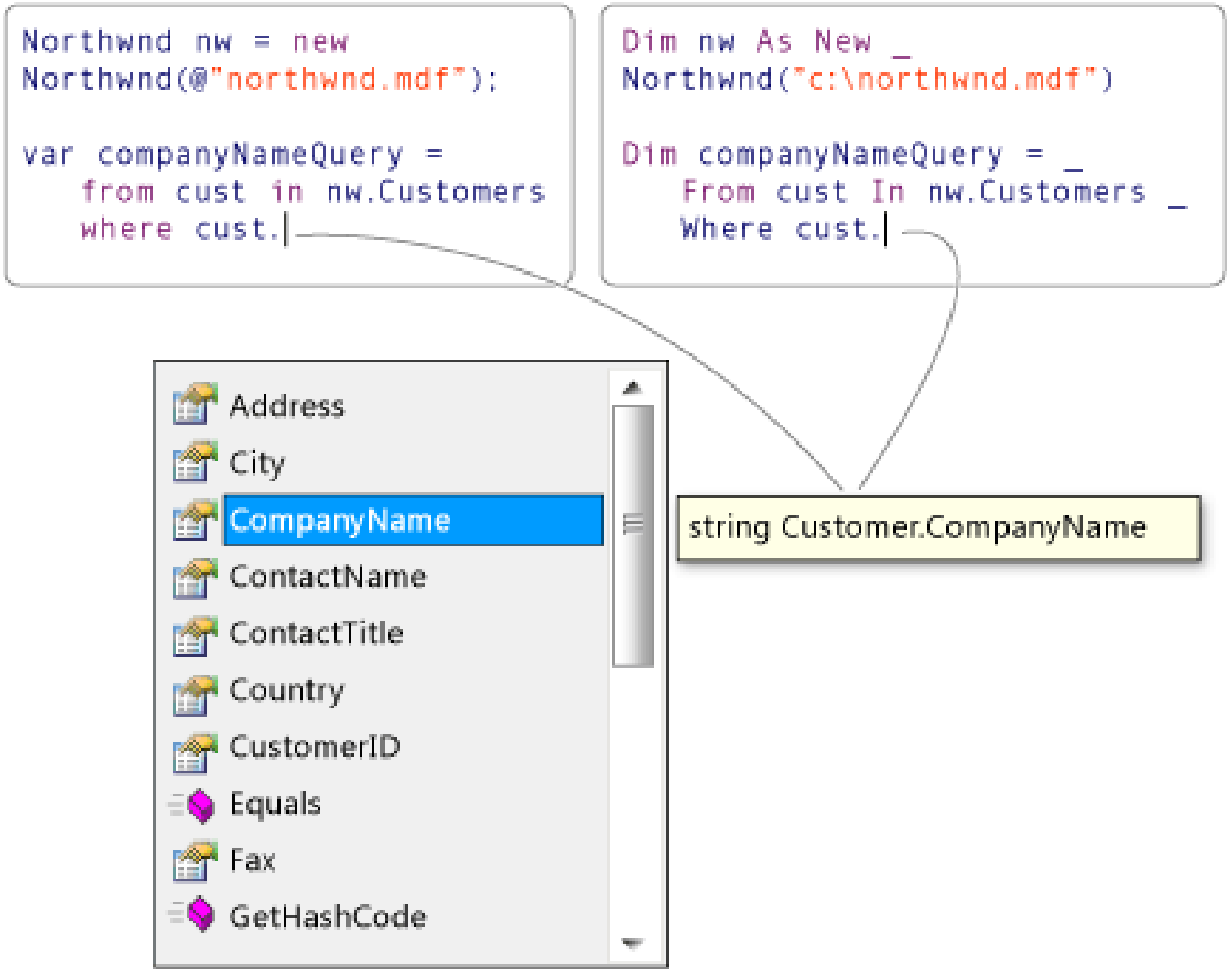
https://en.wikipedia.org/wiki/List_of_object-relational_mapping_software

LINQ to SQL

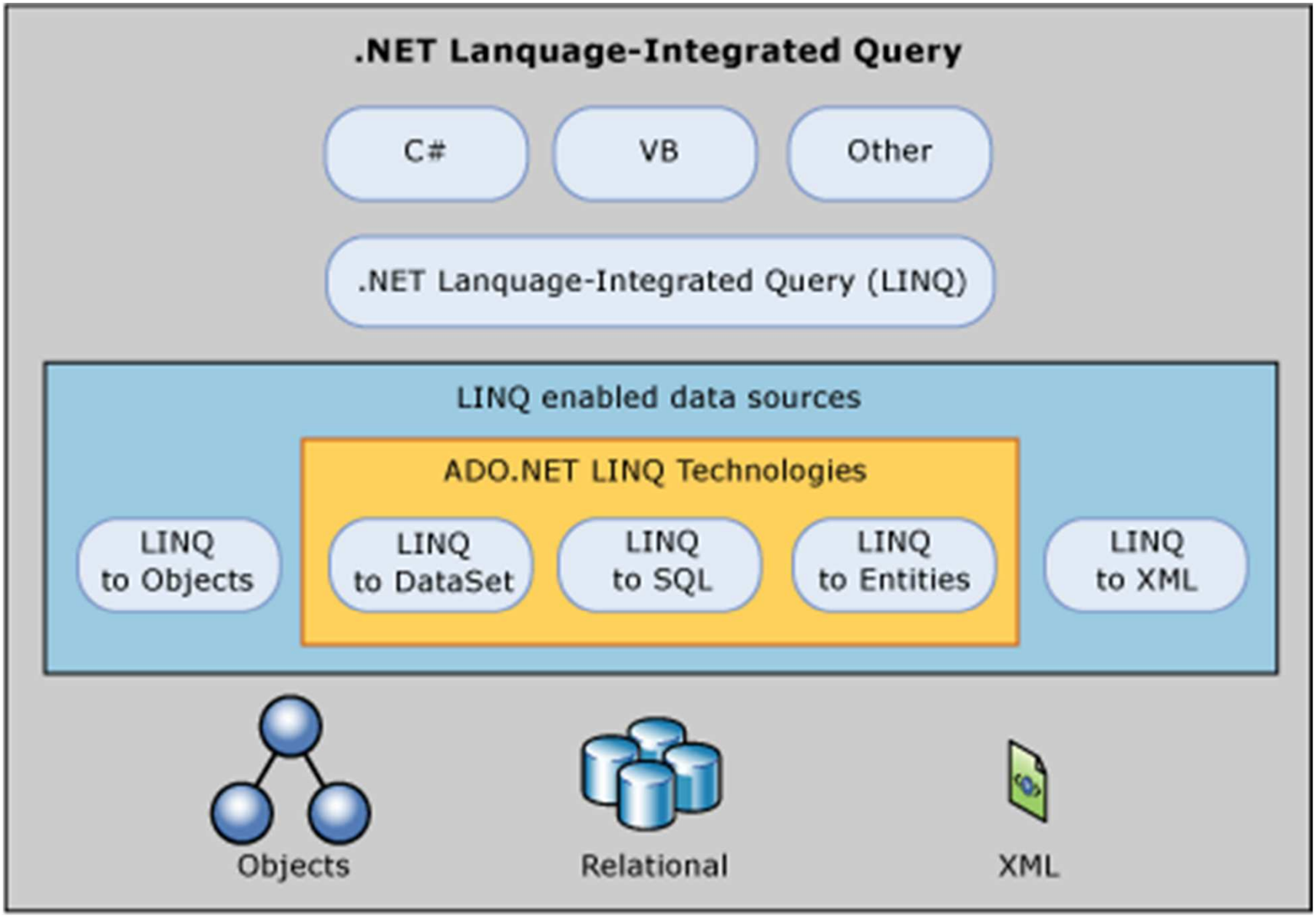
- LINQ extends the language by the addition of query expressions, which are akin to SQL statements
- Can be used to conveniently extract and process data from arrays, enumerable classes, XML documents, relational databases, and third-party data sources

LINQ

- A way of OR mapping : Bridges the gap between the world of objects and the world of data



LINQ and ADO.NET



LINQ to SQL

```
// Northwnd inherits from System.Data.Linq.DataContext.
Northwnd nw = new Northwnd(@"northwnd.mdf");
// or, if you are not using SQL Server Express
// Northwnd nw = new Northwnd("Database=Northwind;Server=server_name;

var companyNameQuery =
    from cust in nw.Customers
    where cust.City == "London"
    select cust.CompanyName;

foreach (var customer in companyNameQuery)
{
    Console.WriteLine(customer);
}
```

http://www.aspsnippets.com/Articles/Simple-Tutorial-with-example-of-using-LINQ-to-SQL-in-ASPNet-Website-using-C-and-VBNet.aspx (For Northwind LINQ Sample)

Connecting to DBMS from Python

- Most common databases for Python web apps
  - PostgreSQL database
    - Using psycopg2
  - MySQL database
    - Using MySQLdb
- Object-relational Mapping
  - Allow developers to access data from a backend by writing Python code instead of SQL queries
  - More on next slide
- Database third-party services
  - Amazon Relational Database Service (RDS) provides pre-configured MySQL and PostgreSQL instances
  - Google Cloud SQL is a service with managed, backed up, replicated, and auto-patched MySQL instances

- See https://www.fullstackpython.com/databases.html

ORM for Python

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...	...	...	...

Python objects

```
class Person:
  first_name = "John"
  last_name = "Connor"
  phone_number = "+16105551234"
```

```
class Person:
  first_name = "Matt"
  last_name = "Makai"
  phone_number = "+12025555689"
```

```
class Person:
  first_name = "Sarah"
  last_name = "Smith"
  phone_number = "+19735554512"
```

ORMs provide a bridge between **relational database tables, relationships and fields** and **Python objects**

web framework	None	Flask	Flask	Django
ORM	SQLAlchemy	SQLAlchemy	SQLAlchemy	Django ORM
database connector	(built into Python stdlib)	MySQL-python	psycopg	psycopg
relational database	SQLite	MySQL	PostgreSQL	PostgreSQL

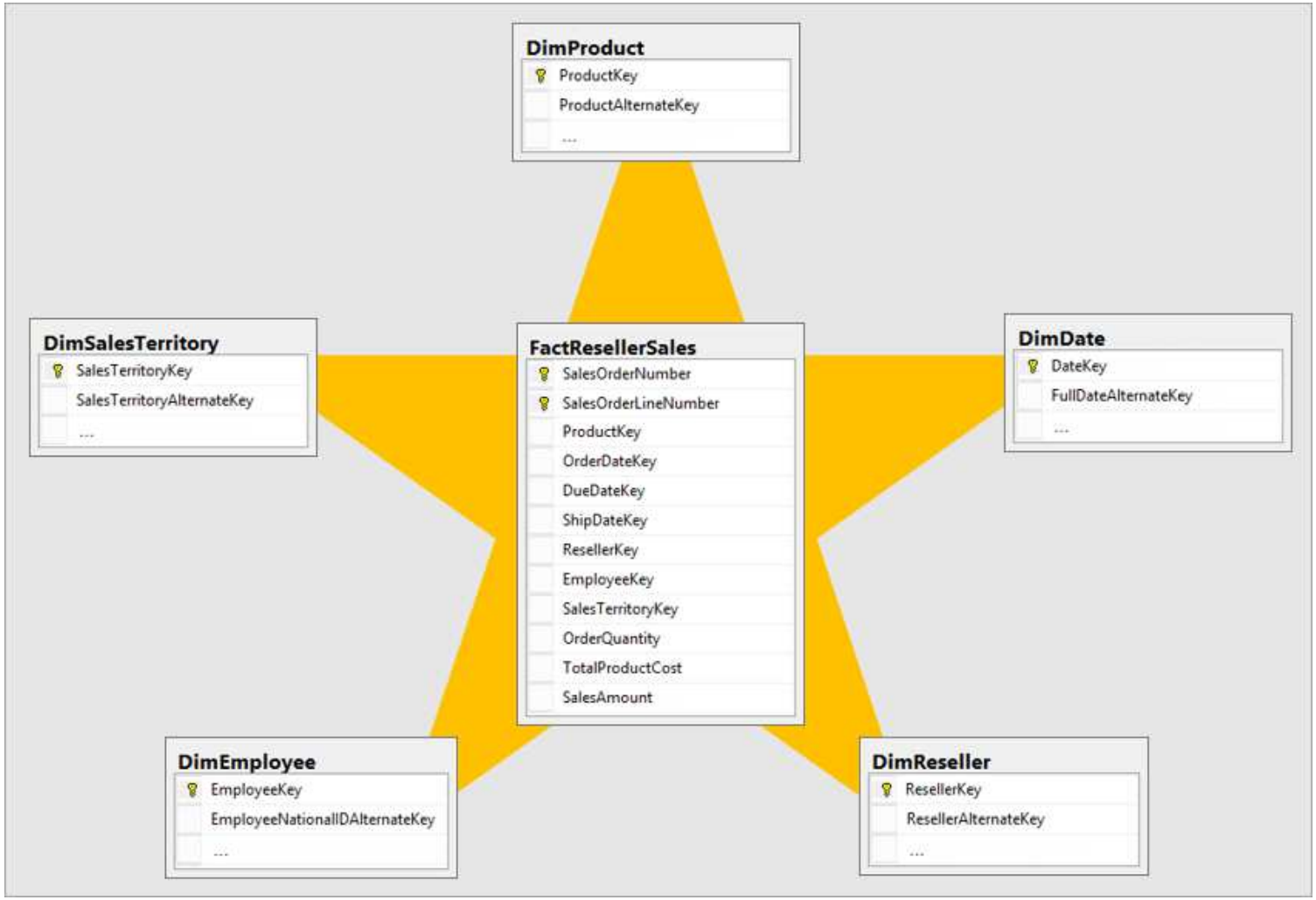
Python+Postgre Databases learning checklist

1) Install PostgreSQL on your server.
2) Make sure the psycopg2 library is in your application's dependencies.
3) Configure your web application to connect to the PostgreSQL instance.
4) Create models in your ORM, either with Django's built-in ORM or SQLAlchemy with Flask.
5) Build your database tables or sync the ORM models with the PostgreSQL instance, if you're using an ORM.
6) Start creating, reading, updating and deleting data in the database from your web application

Columnar Storage for OLAP

Why column store? Data warehouse and Star schema

- Star schema (for OLAP): Facts and Dimensions
- Access patterns: Mostly columnar



https://docs.microsoft.com/en-us/power-bi/guidance/star-schema

Example query on star schema

```
USE [AdventureworksDW2016CTP3]
GO
SELECT * into FactResellerSalesXL From FactResellerSaleXL_CCI
USE [AdventureworksDW2016CTP3]
GO
SET ANSI_PADDING ON
GO
ALTER TABLE [dbo].[FactResellerSalesXL] ADD
CONSTRAINT [PK_FactResellerSalesXL_SalesOrderNumber_SalesOrderLineNumber]
PRIMARY KEY CLUSTERED
(
    [SalesOrderNumber] ASC,
    [SalesOrderLineNumber] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
  SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
```

```
USE [AdventureworksDW2016CTP3]
GO
SET STATISTICS IO ON
GO
SET STATISTICS TIME ON;
GO
SELECT ProductKey, sum(SalesAmount) SalesAmount, sum(OrderQuantity) ct
FROM dbo.FactResellerSalesXL
GROUP BY ProductKey
```

- AdventureWorks DWH is the star schema version of AdventureWorks (for data warehouse)
- The above table has 11.6 million rows
- We will compare query execution plan with/without Columnstore Indexes

Columnstore Indexes

- The standard for storing and querying large data warehousing fact tables
- Uses ***column-based data storage*** and query processing
- Gains up to 10 times the query performance over traditional row-oriented storage

Storage methods

- Columnstore index: a technology for storing, retrieving, and managing data by using a columnar data format, called a columnstore

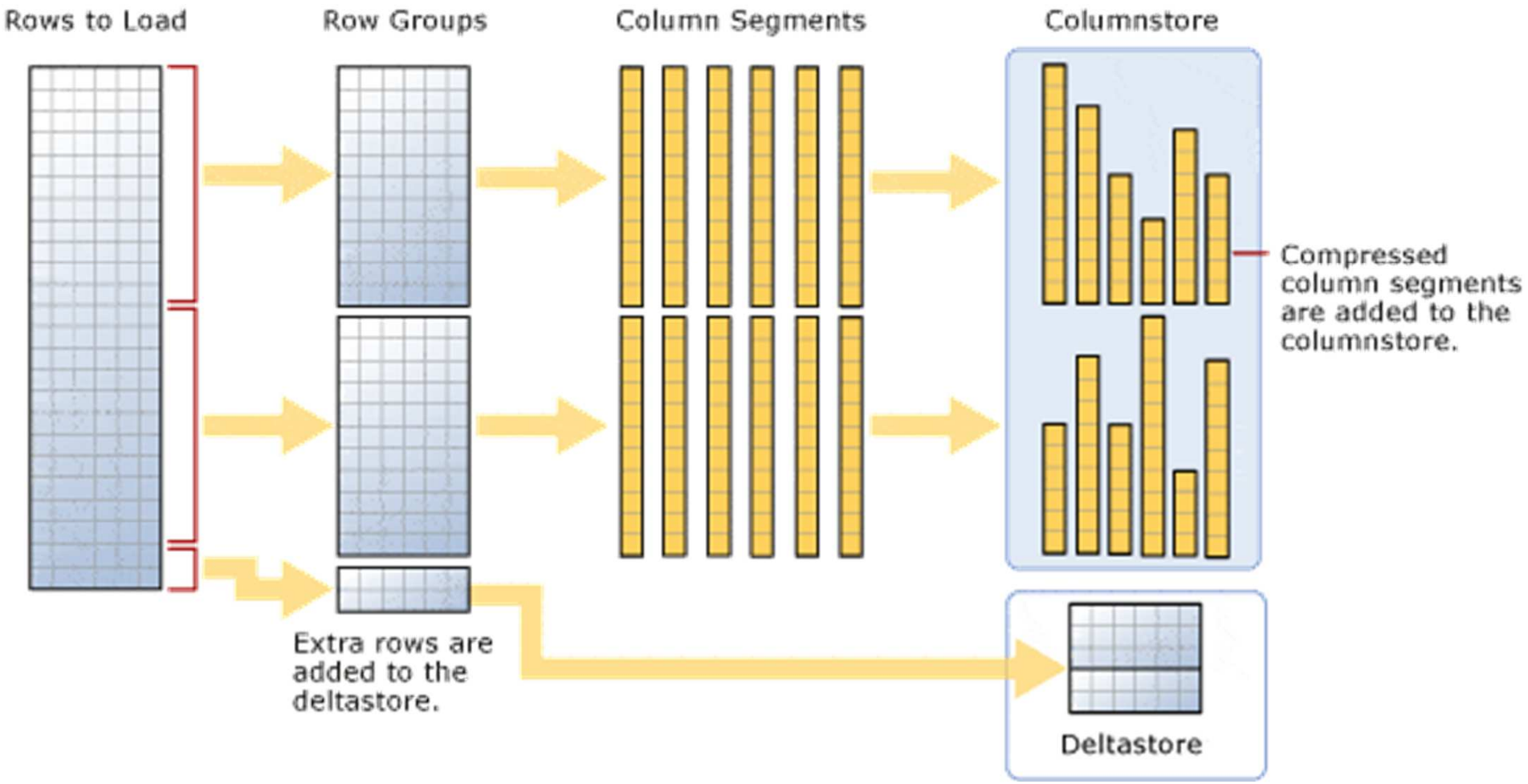
- Rowstore: Data that's **logically** organized as a table with rows and columns, and **physically** stored in a row-wise data format
  - The traditional way to store relational table data
  - Slow OLAP queries

- Columnstore: data that's **logically** organized as a table with rows and columns, and **physically** stored in a column-wise data format

Rowgroup

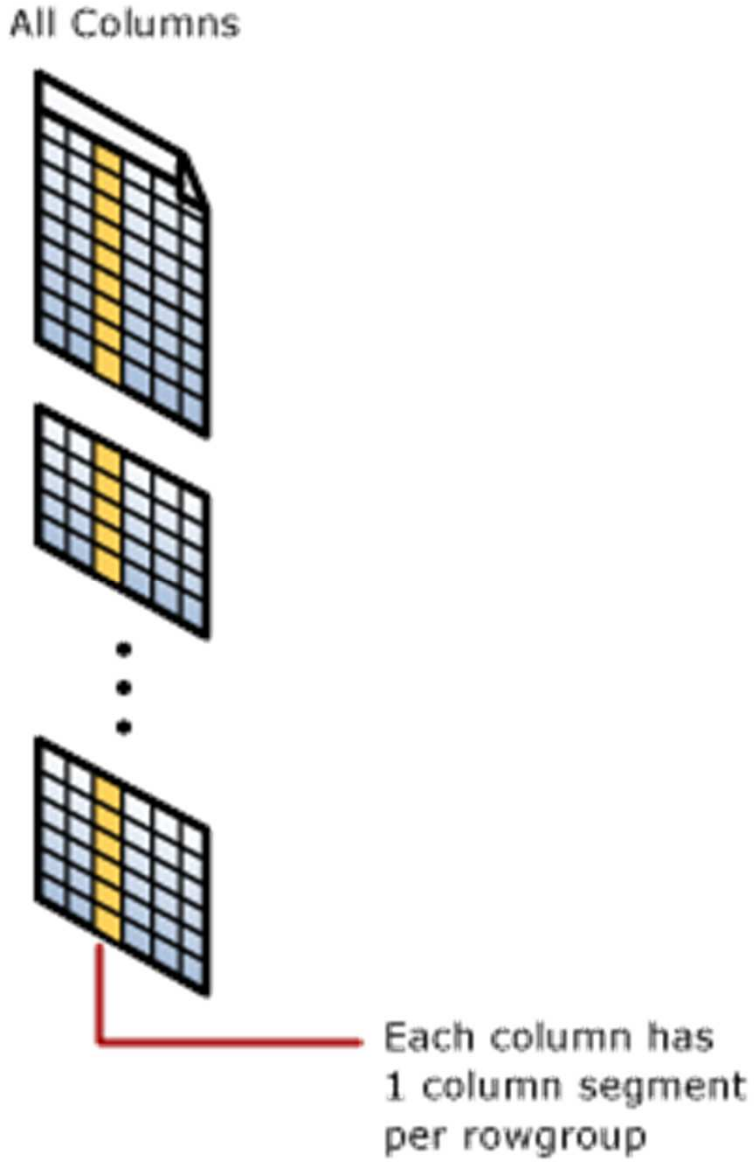
- Rowgroup: a group of rows that are **compressed into columnstore** format at the same time
  - Usually contains the maximum number of rows per rowgroup, which is 1,048,576 rows
- For high **performance** and high **compression** rates, the columnstore index slices the table into rowgroups, and then compresses each rowgroup in a **column-wise manner**
- The number of rows in the rowgroup must be:
  - **large** enough to improve compression rates
  - **small** enough to benefit from in-memory operations

Architecture of Columnstore Indexes



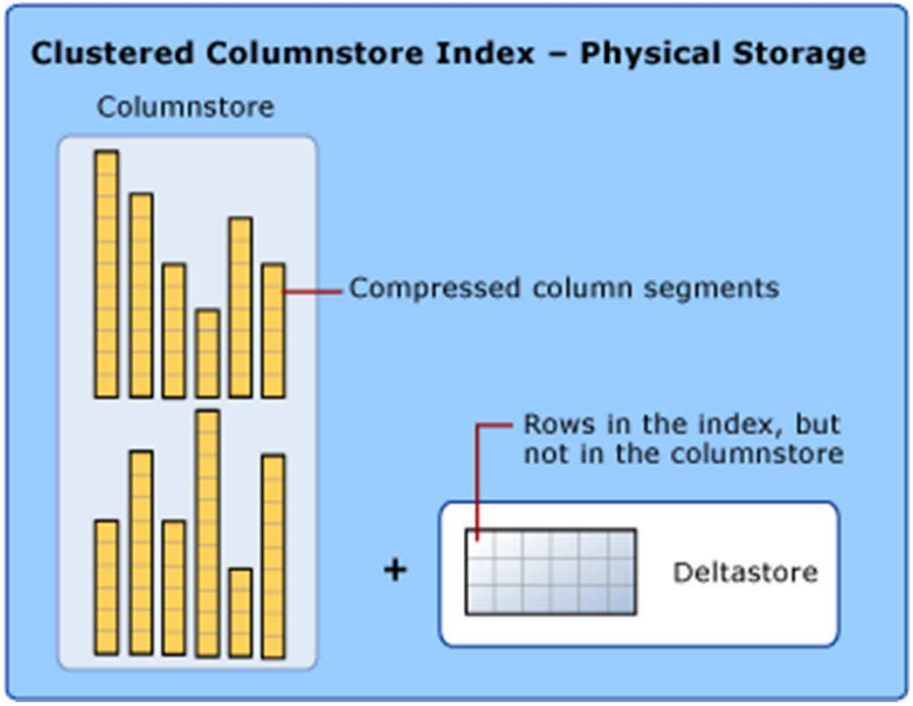
Column segment

- A column segment is a column of data from within the rowgroup
- Each rowgroup contains one column segment for every column in the table
- Each column segment is compressed together and stored on physical media



Clustered columnstore index

- The physical storage for the entire table
- To reduce fragmentation of the column segments and improve performance, the columnstore index might store some data temporarily into a clustered index called a deltastore and a btree list of IDs for deleted rows
- The deltastore operations are handled behind the scenes
- To return the correct query results, the clustered columnstore index combines query results from both the ***columnstore*** and the ***deltastore***



- Starting with SQL Server 2016 (13.x), you can use columnstore indexes for real-time analytics on your operational workload

https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-architecture?view=sql-server-ver15

Comparing execution plans

Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	11669638
Actual Number of Rows	11669638
Actual Number of Batches	0
Estimated Operator Cost	240.048 (90%)
Estimated I/O Cost	233.63
Estimated CPU Cost	6.41838
Estimated Subtree Cost	240.048
Number of Executions	4
Estimated Number of Executions	1
Estimated Number of Rows to be Read	11669600
Estimated Number of Rows	11669600
Estimated Row Size	21 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	7

Object
[AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL].[PK_FactResellerSalesXL_SalesOrderNumber_SalesOrderLineNumber]

Output List
[AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL].ProductKey, [AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL].OrderQuantity, [AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL].SalesAmount

Columnstore Index Scan (Clustered)

Scan a columnstore index, entirely or only a range.

Physical Operation	Columnstore Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Batch
Estimated Execution Mode	Batch
Storage	ColumnStore
Actual Number of Rows	0
Actual Number of Batches	0
Estimated Operator Cost	1.58719 (36%)
Estimated I/O Cost	0.945347
Estimated CPU Cost	0.641838
Estimated Subtree Cost	1.58719
Number of Executions	4
Estimated Number of Executions	1
Estimated Number of Rows	11669600
Estimated Number of Rows to be Read	11669600
Estimated Row Size	21 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Actual Number of Locally Aggregated Rows	11669638
Node ID	3

Object
[AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL_CCI].[IndFactResellerSalesXL_CCI]

Output List
[AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL_CCI].ProductKey, [AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL_CCI].OrderQuantity, [AdventureworksDW2016CTP3].[dbo].[FactResellerSalesXL_CCI].SalesAmount

https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming-sql-server/what-are-columnstore-indexes/

Other column store databases

Database Name	Language Implemented in	Notes
Apache Druid	Java	started in 2011 for low-latency massive ingestion and queries
Apache Kudu	C++	released in 2016 to complete the Apache Hadoop ecosystem
Apache Pinot	Java	open sourced in 2015 for real-time low-latency analytics
Calpont InfiniDB	C++	
ClickHouse	C++	released in 2016 to analyze data that is updated in real time
CrateDB	Java	
C-Store		
DuckDB	C++	An embeddable, in-process, column-oriented SQL OLAP RDBMS
Databend	Rust	An elastic and reliable Serverless Data Warehouse
InfluxDB	Go	time series database
Greenplum Database	C	
PostgreSQL cstore fdw,[1] vops[2]	C	cstore_fdw uses ORC format
MariaDB ColumnStore	C & C++	formerly Calpont InfiniDB
MapD	C++	
Metakit	C++	
MonetDB	C	

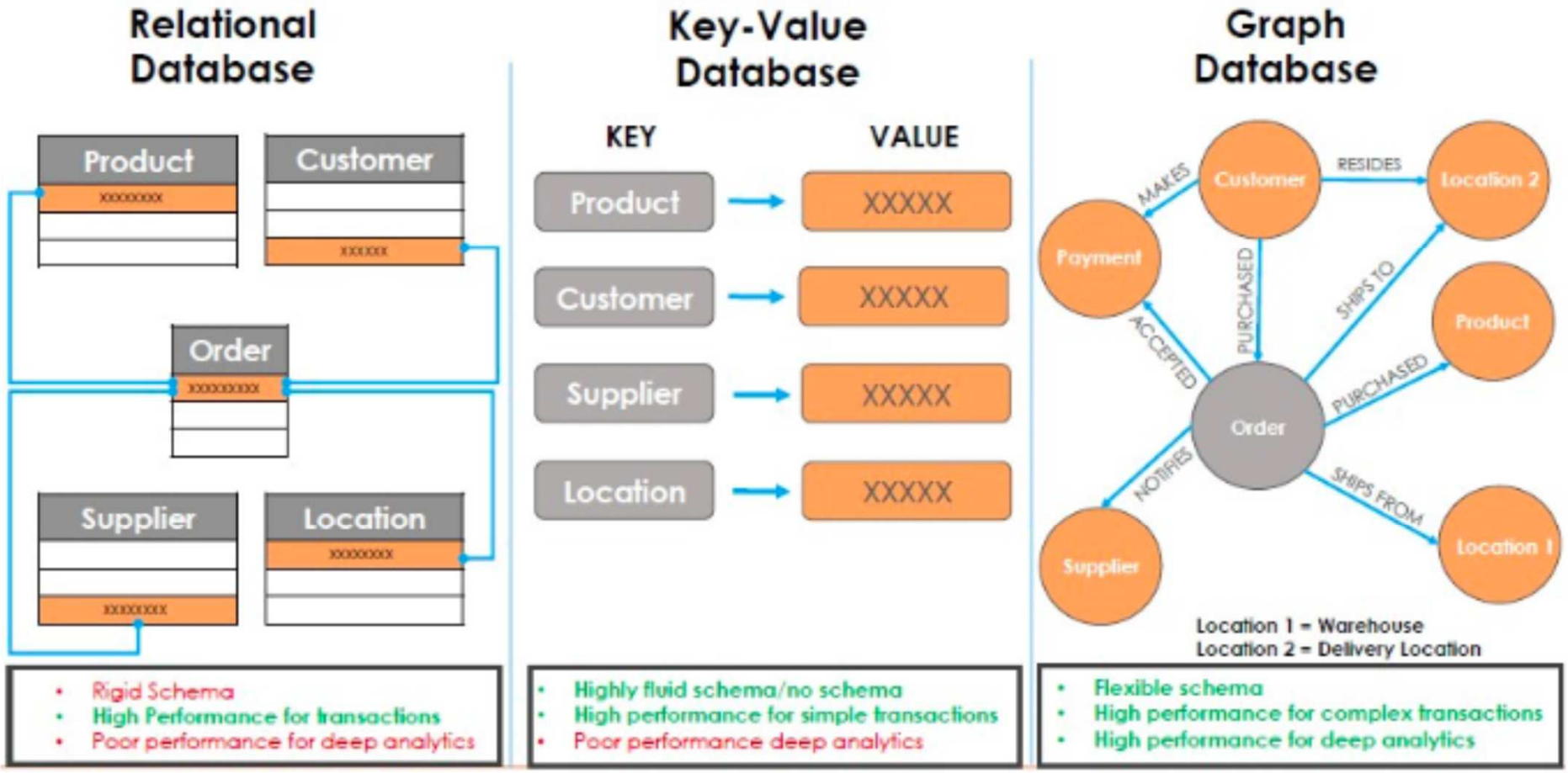
https://en.wikipedia.org/wiki/List_of_column-oriented_DBMSes

When to use column store

- Benefits:
  - **Reduced IO** if only some attributes are accessed
  - Improved **CPU cache** performance
  - Improved **compression**
  - **Vector processing** on modern CPU architectures
- Drawbacks
  - Cost of tuple reconstruction from columnar representation
  - Cost of tuple deletion and update
  - Cost of decompression
- Columnar representation found to be more efficient for decision support than row-oriented representation
- Traditional row-oriented representation preferable for transaction processing
- Some databases support both representations
  - Called hybrid row/column stores

Graph Databases: Modeling emerging data domains

Why graph database?



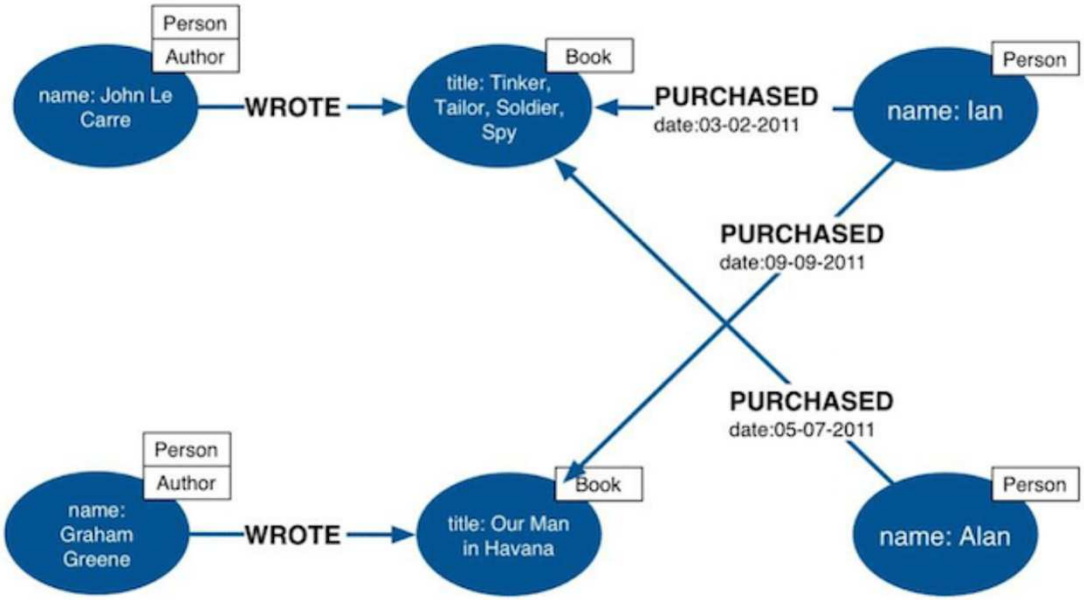
- Different models for different applications

Graph Database Applications

- Social Networks
- Fraud Detection
- 360 Customer Views
- Recommendation Engines
- Network/Operations Mapping
- AI Knowledge Graphs
- Supply Chain Mapping
- Genetics
- Natural Language Processing

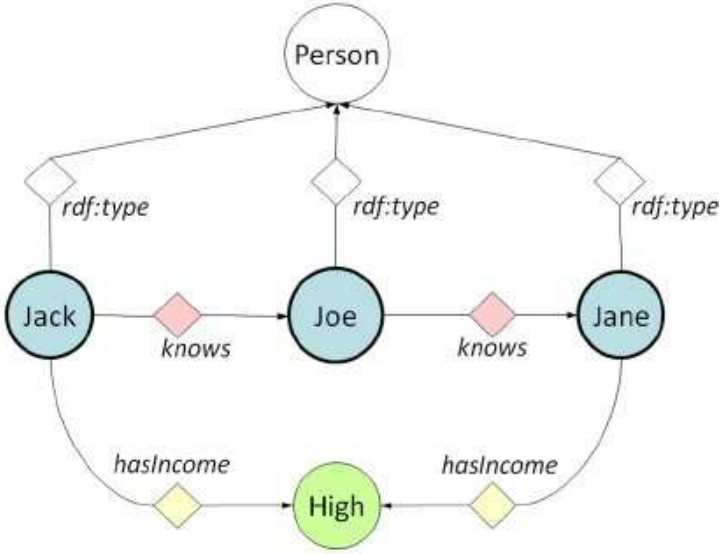
Graph models: Labeled-property graph

- **Labeled-property graph**
- Represented by a set of nodes, relationships, properties, and labels
- Both nodes of data and their relationships are named and can store properties represented by key–value pairs
- Nodes can be labelled to be grouped
- The edges representing the relationships have two qualities
  - A **start** node
  - An **end** node
  - Directed
- **Relationships** can also have properties
  - Useful in providing additional metadata and semantics to relationships of the nodes.
- Direct storage of relationships allows a constant-time traversal



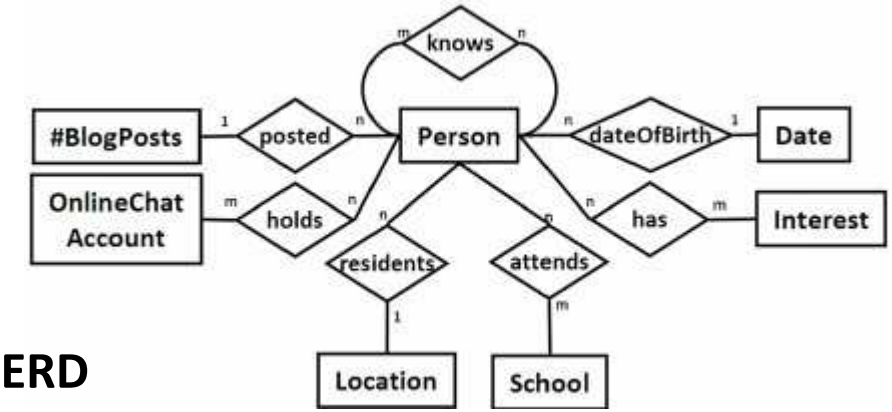
Graph models : RDF

- **Resource Description Framework (RDF):** addition of information is each represented with a **separate node**
- A scenario where a user has to add a **name property** for a person represented as a distinct node in the graph
  - In a labeled-property graph model: addition of a name property into the node of the person
  - In an RDF: add a separate node called hasName connecting it to the original person node
    - Specifically, an RDF graph model is composed of nodes and arcs.



Vertices
Resources : URIs
Attribute Values : Literal Values
Edges
Relationships : URIs

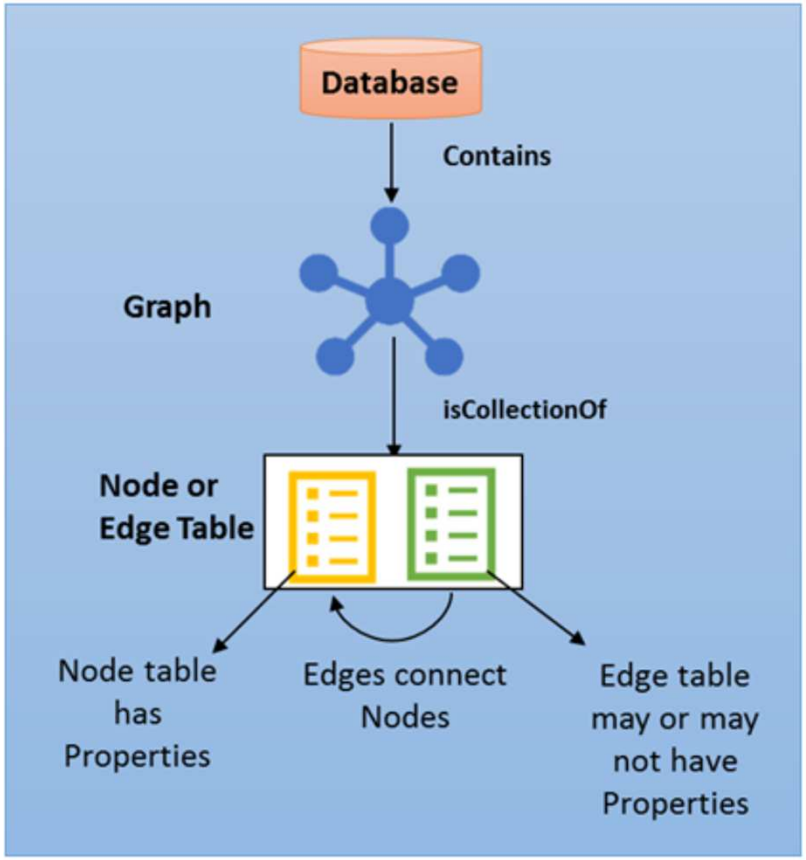
Nodes or Edges have NO internal structure



ERD

SQL Graph Database

- A graph is a collection of node and edge tables
- Node or edge tables can be created under any schema in the database, but they all belong to one logical graph
- A node table is collection of similar type of nodes
  - For example, a Person node table holds all the Person nodes belonging to a graph
- An edge table is a collection of similar type of edges
  - For example, a Friends edge table holds all the edges that connect a Person to another Person
- Since nodes and edges are stored in tables, most of the operations supported on regular tables are supported on node or edge tables



Node and edge table representation

- Stored as tables

Node Properties

$node_id	Name	Age
{"type":"node","id":0}	John	30
"type":"node","id":1}	Mary	28
"type":"node","id":2}	Alice	25

Person Node Table

Nodes that this edge connects

Edge Properties

$edge_id	$from_id	$to_id	StartDate
{"type":"edge","id":0}	{"type":"node","id":0}	{"type":"node","id":1}	01/01/2013
{"type":"edge","id":1}	{"type":"node","id":1}	{"type":"node","id":2}	05/05/2010
{"type":"edge","id":2}	{"type":"node","id":2}	{"type":"node","id":0}	09/09/2016

Friends Edge Table

https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-architecture?view=sql-server-ver15

Querying graph databases

- **MATCH** command for pattern matching

A. Find a friend

The following example creates a Person node table and friends Edge table, inserts some data and then uses MATCH to find friends of Alice, a person in the graph.

SQL Copy

```
-- Create person node table
CREATE TABLE dbo.Person (ID INTEGER PRIMARY KEY, name VARCHAR(50)) AS NODE;
CREATE TABLE dbo.friend (start_date DATE) AS EDGE;

-- Insert into node table
INSERT INTO dbo.Person VALUES (1, 'Alice');
INSERT INTO dbo.Person VALUES (2,'John');
INSERT INTO dbo.Person VALUES (3, 'Jacob');

-- Insert into edge table
INSERT INTO dbo.friend VALUES ((SELECT $node_id FROM dbo.Person WHERE name = 'Alice'),
       (SELECT $node_id FROM dbo.Person WHERE name = 'John'), '9/15/2011');

INSERT INTO dbo.friend VALUES ((SELECT $node_id FROM dbo.Person WHERE name = 'Alice'),
       (SELECT $node_id FROM dbo.Person WHERE name = 'Jacob'), '10/15/2011');

INSERT INTO dbo.friend VALUES ((SELECT $node_id FROM dbo.Person WHERE name = 'John'),
       (SELECT $node_id FROM dbo.Person WHERE name = 'Jacob'), '10/15/2012');

-- use MATCH in SELECT to find friends of Alice
SELECT Person2.name AS FriendName
FROM Person Person1, friend, Person Person2
WHERE MATCH(Person1-(friend)->Person2)
AND Person1.name = 'Alice';
```

B. Find friend of a friend

The following example tries to find friend of a friend of Alice.

SQL Copy

```
SELECT Person3.name AS FriendName
FROM Person Person1, friend, Person Person2, friend friend2, Person Person3
WHERE MATCH(Person1-(friend)->Person2-(friend2)->Person3)
AND Person1.name = 'Alice';
```

An application in healthcare

A Graph Database Approach for Temporal Modeling of Disease Progression

Hoda Memarzadeh[1], Nasser Ghadiri[1], Sara Parikhah Zarmehr[1]
[1]*Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*
{ h.memarzadeh@ec.iut.ac.ir, nghadiri@cc.iut.ac.ir, s.parikhah@ec.iut.ac.ir }

***Abstract*— The high cost of managing chronic diseases for individuals and governments, as well as the negative impact on the quality of life, highlights the importance of controlling and**

among patients but little is known about prognostic predictors[4].
There are many different approaches for statistical modeling of

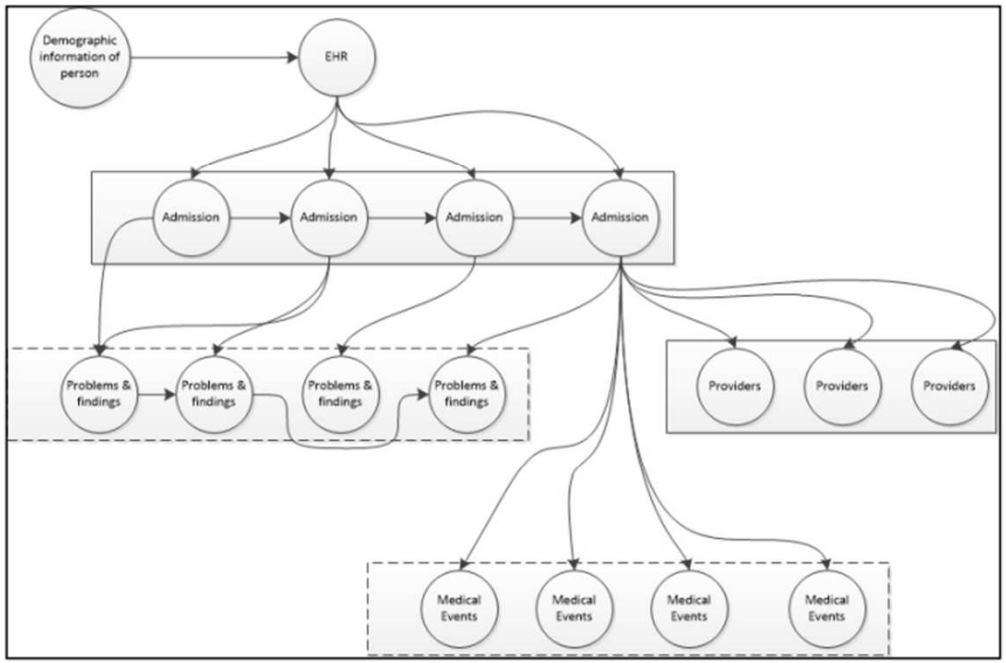


Fig. 7. An extended example of graph-based design for EHR

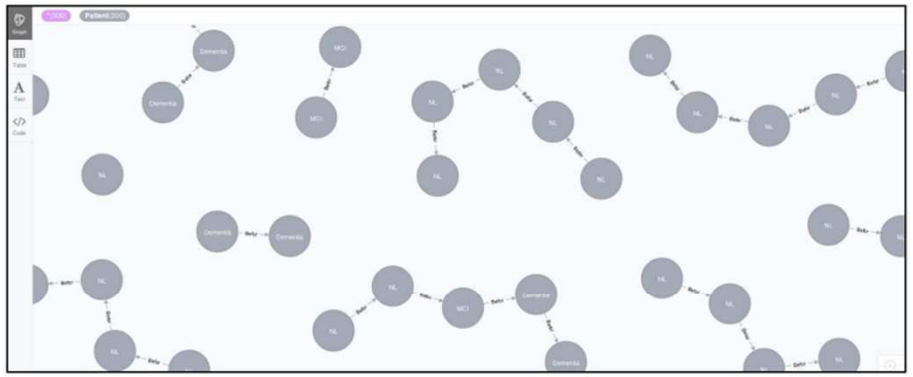


Fig. 2. A view of multiple sequences of transactions for a number of patients with different lengths.

```
MATCH (Start:Patient )-[r:Befor*1..10]->(End:Patient)
return Start.DX,End.Year-Start.Year ,End.DX, count(r)
```

Fig. 3. Match command for searching paths

Aftcr cxccuting this command as shown in "Fig. 4" all thc different paths are distinguished between the two stages of disease by the time interval are shown based on the observed number of that path.

"Start.DX"	"End.Year-Start.Year"	"End.DX"	"count(r)"
"NL"	0	"NL"	218
"NL"	4	"Dementia"	19
"Dementia"	3	"MCI"	2
"Dementia"	4	"Dementia"	33
"MCI"	1	"Dementia"	182
"MCI"	5	"MCI"	141
"MCI"	6	"Dementia"	6
"NL"	5	"MCI"	19
"Dementia"	0	"MCI"	4
"Dementia"	3	"NL"	1
"NL"	1	"Dementia"	3

Fig. 4. Sample of result of Match command execution

Other graph databases

- Amazon Neptune
  - Fully managed graph database by Amazon.com
- Neo4j
  - Open-source, supports ACID, has high-availability clustering for enterprise deployments
- Ontotext GraphDB
  - Highly efficient and robust graph database with RDF and SPARQL support, also available as a high-availability cluster
- Stardog
  - Enterprise knowledge graph platform supporting RDF and labeled property graphs
- A long list at https://en.wikipedia.org/wiki/Graph_database

DBMS Benchmarks: Comparing and Selecting DBMSs

Comparison of relational database management systems

Oracle DB	Oracle Corporation	1979-11	19c[28]	2019-02-13; 2 years ago	Proprietary	No
Oracle Rdb	Oracle Corporation	1984	7.4.1.1[29]	2021-04-21[±]	Proprietary	No
Paradox	Corel Corporation	1985	11	2009-09-07	Proprietary	No
Percona Server for MySQL	Percona	2006	8.0.25-15	2021-07-13[±]	GPL v2	Yes
Pervasive PSQL	Pervasive Software	1982	v12	2015	Proprietary	No
Polyhedra DBMS	ENEA AB	1993	9.0	2015-06-24	Proprietary, with Polyhedra Lite available as Freeware[30]	No
PostgreSQL	PostgreSQL Global Development Group	1989-06	14.1[31]	2021-11-11; 40 days ago	Postgres License[32]	No[33]
R:Base	R:BASE	1982	10.0	2016-05-26	Proprietary	No

https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

DB-Engines (*Popularity)* Ranking

381 systems in ranking, December 2021

Rank Dec 2021	Rank Nov 2021	Rank Dec 2020	DBMS	Database Model	Score Dec 2021	Score Nov 2021	Score Dec 2020
1.	1.	1.	Oracle	Relational, Multi-model	1281.74	+9.01	-43.86
2.	2.	2.	MySQL	Relational, Multi-model	1206.04	-5.48	-49.41
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	954.02	-0.27	-84.07
4.	4.	4.	PostgreSQL	Relational, Multi-model	608.21	+10.94	+60.64
5.	5.	5.	MongoDB	Document, Multi-model	484.67	-2.67	+26.95
6.	6.	↑ 7.	Redis	Key-value, Multi-model	173.54	+2.04	+19.91
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	167.18	-0.34	+6.74
8.	8.	8.	Elasticsearch	Search engine, Multi-model	157.72	-1.36	+5.23
9.	9.	9.	SQLite	Relational	128.68	-1.12	+7.00
10.	↑ 11.	↑ 11.	Microsoft Access	Relational	125.99	+6.75	+9.25
11.	↓ 10.	↓ 10.	Cassandra	Wide column	119.20	-1.68	+0.36
12.	12.	12.	MariaDB	Relational, Multi-model	104.36	+2.17	+10.75
13.	13.	13.	Splunk	Search engine	94.32	+2.02	+7.32
14.	↑ 15.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	83.25	+1.93	+13.76
15.	↓ 14.	15.	Hive	Relational	81.93	-1.38	+11.66
16.	16.	↑ 17.	Amazon DynamoDB	Multi-model	77.63	+0.64	+8.51
17.	↑ 18.	↑ 41.	Snowflake	Relational	71.03	+6.84	+58.12
18.	↓ 17.	↓ 14.	Teradata	Relational, Multi-model	70.29	+0.71	-3.54
19.	19.	19.	Neo4j	Graph	58.03	+0.05	+3.40
20.	↑ 22.	↑ 21.	Solr	Search engine, Multi-model	57.72	+3.87	+6.48
21.	↓ 20.	↓ 20.	SAP HANA	Relational, Multi-model	54.58	-0.95	+2.08

https://db-engines.com/en/ranking

DB-Engines: Example page

PostgreSQL System Properties

Please select another system to compare it with PostgreSQL.

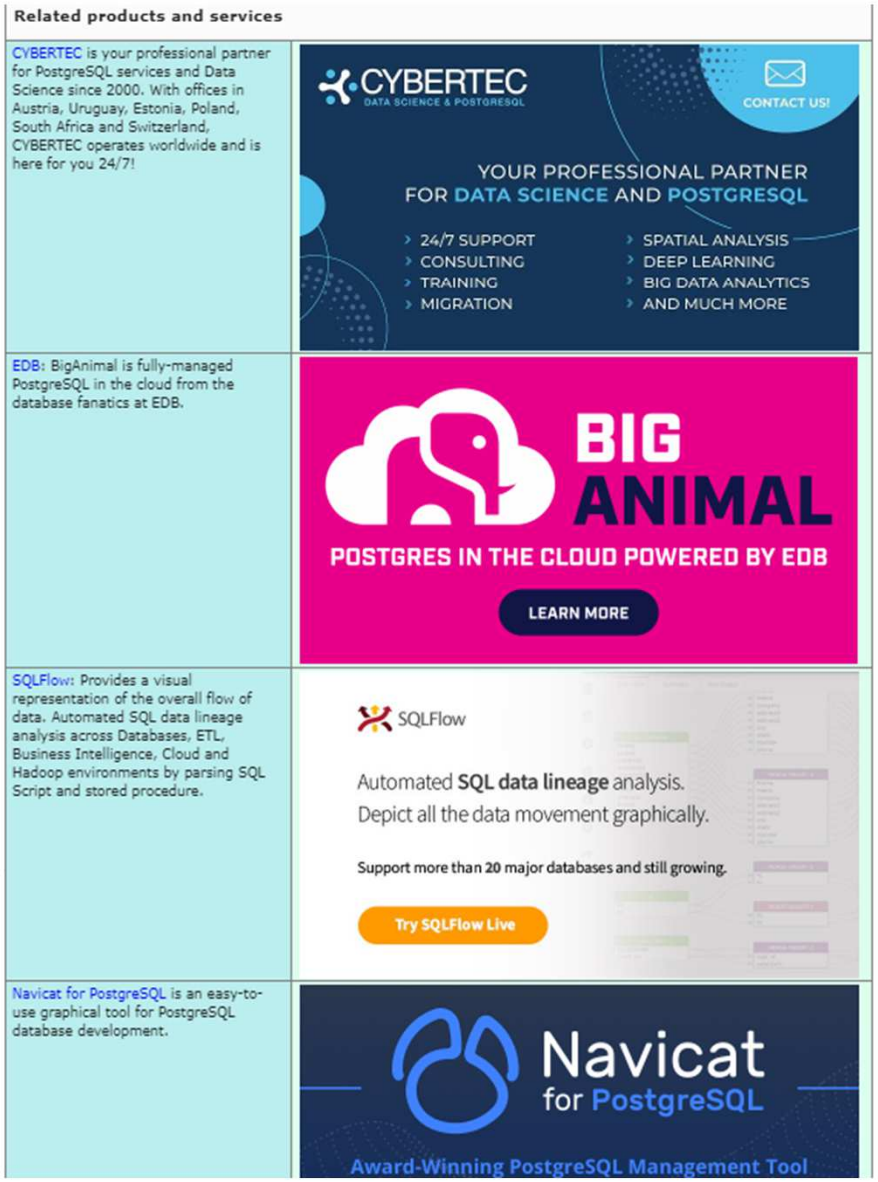
Our visitors often compare PostgreSQL with MySQL, MariaDB and Microsoft SQL Server.

Editorial information provided by DB-Engines

Name	**PostgreSQL**
Description	Widely used open source RDBMS
Primary database model	Relational DBMS
Secondary database models	Document store Spatial DBMS
DB-Engines Ranking Trend Chart	Score 608.21 Rank #4 Overall #4 Relational DBMS
Website	www.postgresql.org
Technical documentation	www.postgresql.org/docs
Developer	PostgreSQL Global Development Group
Initial release	1989
Current release	14.1, November 2021

Server operating systems	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows

Supported programming languages	.Net C C++ Delphi Java JavaScript (Node.js) Perl PHP Python Tcl



https://db-engines.com/en/system/PostgreSQL

TPC DBMS (*Technical*) Benchmarks

http://www.tpc.org/

The Transaction Processing Performance Council defines transaction processing and database benchmarks and delivers trusted results to the industry.

TPC-E

- TPC-E is an On-Line Transaction Processing Benchmark
- TPC-E involves a mix of twelve concurrent transactions of different types and complexity
- Executed on-line or triggered by price or time criteria
- Database is comprised of thirty-three tables
  - with a wide range of columns, cardinality, and scaling properties.

Rank	Company	System	Performance (tpsE)	Price/tpsE	Watts/tpsE	System Availability	Database	Operating System	Processors / Cores / Threads	Date Submitted
1	Lenovo	Lenovo ThinkSystem SR860 V2	12,163	84.96 USD	NR	11/19/20	Microsoft SQL Server 2019 Enterprise Edition	Microsoft Windows Server 2016 Standard Edition	4 / 112 / 224	11/19/20
2	Lenovo	Lenovo ThinkSystem SR665	12,028	91.85 USD	NR	03/18/21	Microsoft SQL Server 2019 Enterprise Edition	Microsoft Windows Server 2019 Standard Edition	2 / 128 / 256	03/11/21
3	Lenovo	Lenovo ThinkSystem SR655	7,891	76.92 USD	NR	06/15/21	Microsoft SQL Server 2019 Enterprise Edition	Microsoft Windows Server 2016 Standard Edition	1 / 64 / 128	06/04/21
4	Lenovo	Lenovo ThinkSystem SR650	7,013	90.99 USD	NR	04/17/19	Microsoft SQL Server 2017 Enterprise Edition	Microsoft Windows Server 2016 Standard Edition	2 / 56 / 112	03/29/19
5	FUJITSU	Fujitsu Server PRIMERGY RX2540 M5	6,844	85.13 USD	NR	10/24/19	Microsoft SQL Server 2017 Enterprise Edition	Microsoft Windows Server 2016 Standard Edition	2 / 56 / 112	10/23/19
6	Lenovo	Lenovo ThinkSystem SR655	6,717	99.99 USD	NR	12/31/19	Microsoft SQL Server 2017 Enterprise Edition	Microsoft Windows Server 2016 Standard Edition	1 / 64 / 128	08/02/19
7	Lenovo	Lenovo ThinkSystem SR665	2,579	68.62 USD	NR	08/17/21	Microsoft SQL Server 2019 Enterprise Edition	Microsoft Windows Server 2019 Standard Edition	2 / 16 / 32	08/12/21

TPC-H

- TPC-H is illustrates **decision support** systems that examine **large** volumes of data
- Execute queries with a high degree of complexity, and give answers to critical business questions
- Consists of a suite of business oriented ad-hoc queries and concurrent data modifications
- The queries and the data populating the database have been chosen to have broad industry-wide relevance

100 GB Results

Rank	Company	System	QphH	Price/kQphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Cluster
1	TTA	KTNF KR580S1	43,903	1,567,768.70 KRW	NR	02/16/21	Altibase 7.1	Red Hat Enterprise Linux 7.9	02/15/21	N

1,000 GB Results

Rank	Company	System	QphH	Price/kQphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Clu
1	Hewlett Packard Enterprise	HPE DL325 Gen10	6,145,628	50.40 USD	NR	08/26/19	EXASOL 6.2	CentOS 7.6	07/31/19	
2	Hewlett Packard Enterprise	HPE DL325 Gen10	3,635,443	57.91 USD	NR	08/26/19	EXASOL 6.2	CentOS 7.6	07/31/19	
3	DELL	Dell PowerEdge R7515	979,335	269.23 USD	NR	05/03/21	Microsoft SQL Server 2019 Enterprise Edition 64 bit	Red Hat Enterprise Linux 8	05/03/21	
4	DELL	PowerEdge MX740c Server	824,693	459.50 USD	NR	03/03/21	Microsoft SQL Server 2019 Enterprise Edition	Red Hat Enterprise Linux 8	03/03/21	
5	Hewlett Packard Enterprise	HPE ProLiant DL325 Gen10	743,750	339.21 USD	NR	08/07/19	Microsoft SQL Server 2017 Enterprise Edition	Red Hat Enterprise Linux 8	08/05/19	

3,000 GB Results

Rank	Company	System	QphH	Price/kQphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted	Clu
1	DELL	Dell PowerEdge R6525	7,696,073	68.34 USD	NR	10/22/19	EXASOL 6.2	CentOS 7.6	10/18/19	
2	DELL	Dell PowerEdge R6415	6,053,020	69.50 USD	NR	07/09/19	EXASOL 6.2	CentOS 7.6	07/09/19	