# Graph processing with SQL Server

starting with 2017

SQL Server offers graph database capabilities to model many-to-many relationships. The graph relationships are integrated into Transact-SQL and receive the benefits of using SQL Server as the foundational database management system.

## What is a graph database?

A graph database is a collection of nodes (or vertices) and edges (or relationships). A node represents an entity (for example, a person or an organization) and an edge represents a relationship between the two nodes that it connects (for example, likes or friends). Both nodes and edges may have properties associated with them. Here are some features that make a graph database unique:

* Edges or relationships are first class entities in a Graph Database and can have attributes or properties associated with them.
* A single edge can flexibly connect multiple nodes in a Graph Database.
* You can express pattern matching and multi-hop navigation queries easily.
* You can express transitive closure and polymorphic queries easily.

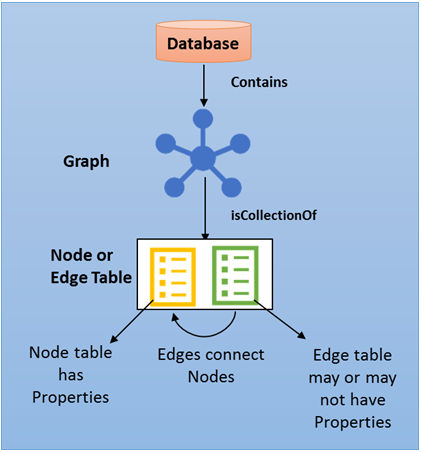
## When to use a graph database

There is nothing a graph database can achieve, which cannot be achieved using a relational database. However, a graph database can make it easier to express certain kind of queries. Also, with specific optimizations, certain queries may perform better. Your decision to choose one over the other can be based on following factors:

* Your application has hierarchical data. The HierarchyID datatype can be used to implement hierarchies, but it has some limitations. For example, it does not allow you to store multiple parents for a node.
* Your application has complex many-to-many relationships; as application evolves, new relationships are added.
* You need to analyze interconnected data and relationships.

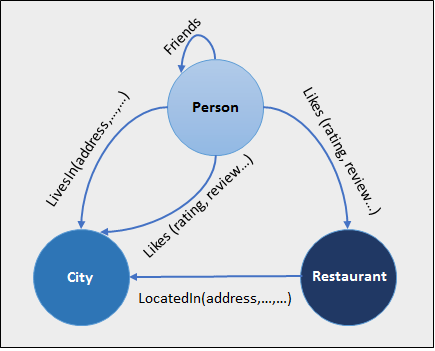
# SQL Graph Architecture

Users can create one graph per 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. Similarly, 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.



Sample Schema

This sample creates a graph schema, as showed in Figure 1, for a hypothetical social network that has People, Restaurant and City nodes. These nodes are connected to each other using Friends, Likes, LivesIn and LocatedIn edges.

  
Figure 1: Sample schema with restaurant, city, person nodes and LivesIn, LocatedIn, Likes edges.

Sample Script

-- Create a graph demo database

CREATE DATABASE graphdemo;

-- Create NODE tables

CREATE TABLE Person ( ID INTEGER PRIMARY KEY, name VARCHAR(100) ) AS NODE;

CREATE TABLE Restaurant ( ID INTEGER NOT NULL, name VARCHAR(100),

city VARCHAR(100) ) AS NODE;

CREATE TABLE City ( ID INTEGER PRIMARY KEY, name VARCHAR(100),

stateName VARCHAR(100)) AS NODE;

-- Create EDGE tables.

CREATE TABLE likes (rating INTEGER) AS EDGE;

CREATE TABLE friendOf AS EDGE;

CREATE TABLE livesIn AS EDGE;

CREATE TABLE locatedIn AS EDGE;

-- Insert data into node tables. Inserting into a node table is same as inserting into a regular table

INSERT INTO Person VALUES (1,'John');INSERT INTO Person VALUES (2,'Mary');

INSERT INTO Person VALUES (3,'Alice');INSERT INTO Person VALUES (4,'Jacob');

INSERT INTO Person VALUES (5,'Julie');

INSERT INTO Restaurant VALUES (1,'Taco Dell','Bellevue');

INSERT INTO Restaurant VALUES (2,'Ginger and Spice','Seattle');

INSERT INTO Restaurant VALUES (3,'Noodle Land', 'Redmond');

INSERT INTO City VALUES (1,'Bellevue','wa');

INSERT INTO City VALUES (2,'Seattle','wa');

INSERT INTO City VALUES (3,'Redmond','wa');

-- Insert into edge table. While inserting into an edge table,

-- you need to provide the $node\_id from $from\_id and $to\_id columns.

INSERT INTO likes VALUES ((SELECT $node\_id FROM Person WHERE id = 1),

(SELECT $node\_id FROM Restaurant WHERE id = 1),9);

INSERT INTO likes VALUES ((SELECT $node\_id FROM Person WHERE id = 2),

(SELECT $node\_id FROM Restaurant WHERE id = 2),9);

INSERT INTO likes VALUES ((SELECT $node\_id FROM Person WHERE id = 3),

(SELECT $node\_id FROM Restaurant WHERE id = 3),9);

INSERT INTO likes VALUES ((SELECT $node\_id FROM Person WHERE id = 4),

(SELECT $node\_id FROM Restaurant WHERE id = 3),9);

INSERT INTO likes VALUES ((SELECT $node\_id FROM Person WHERE id = 5),

(SELECT $node\_id FROM Restaurant WHERE id = 3),9);

INSERT INTO livesIn VALUES ((SELECT $node\_id FROM Person WHERE id = 1),

(SELECT $node\_id FROM City WHERE id = 1));

INSERT INTO livesIn VALUES ((SELECT $node\_id FROM Person WHERE id = 2),

(SELECT $node\_id FROM City WHERE id = 2));

INSERT INTO livesIn VALUES ((SELECT $node\_id FROM Person WHERE id = 3),

(SELECT $node\_id FROM City WHERE id = 3));

INSERT INTO livesIn VALUES ((SELECT $node\_id FROM Person WHERE id = 4),

(SELECT $node\_id FROM City WHERE id = 3));

INSERT INTO livesIn VALUES ((SELECT $node\_id FROM Person WHERE id = 5),

(SELECT $node\_id FROM City WHERE id = 1));

INSERT INTO locatedIn VALUES ((SELECT $node\_id FROM Restaurant WHERE id = 1),

(SELECT $node\_id FROM City WHERE id =1));

INSERT INTO locatedIn VALUES ((SELECT $node\_id FROM Restaurant WHERE id = 2),

(SELECT $node\_id FROM City WHERE id =2));

INSERT INTO locatedIn VALUES ((SELECT $node\_id FROM Restaurant WHERE id = 3),

(SELECT $node\_id FROM City WHERE id =3));

-- Insert data into the friendof edge.

INSERT INTO friendof VALUES ((SELECT $NODE\_ID FROM person WHERE ID = 1), (SELECT $NODE\_ID FROM person WHERE ID = 2));

INSERT INTO friendof VALUES ((SELECT $NODE\_ID FROM person WHERE ID = 2), (SELECT $NODE\_ID FROM person WHERE ID = 3));

INSERT INTO friendof VALUES ((SELECT $NODE\_ID FROM person WHERE ID = 3), (SELECT $NODE\_ID FROM person WHERE ID = 1));

INSERT INTO friendof VALUES ((SELECT $NODE\_ID FROM person WHERE ID = 4), (SELECT $NODE\_ID FROM person WHERE ID = 2));

INSERT INTO friendof VALUES ((SELECT $NODE\_ID FROM person WHERE ID = 5), (SELECT $NODE\_ID FROM person WHERE ID = 4));

-- Find Restaurants that John likes

SELECT Restaurant.name FROM Person, likes, Restaurant

WHERE MATCH (Person-(likes)->Restaurant) AND Person.name = 'John';

-- Find Restaurants that John's friends like

SELECT Restaurant.name FROM Person person1, Person person2, likes, friendOf, Restaurant WHERE MATCH(person1-(friendOf)->person2-(likes)->Restaurant)

AND person1.name='John';

-- Find people who like a restaurant in the same city they live in

SELECT Person.name FROM Person, likes, Restaurant, livesIn, City, locatedIn

WHERE MATCH (Person-(likes)->Restaurant-(locatedIn)->City AND Person-(livesIn)->City);