**Data Processing** in simple terms can be defined as collecting data and converting it into useful information and this conversion process involves certain steps. Traditionally, the major form of data was numbers and processing these numerical data was data processing but the entire purpose and meaning of this term changed with the introduction of computer and computer storage systems. And now the most used terminology in the data processing came into use “database”. In this context, data processing is the process of collecting data, storing it in the database and retrieving the necessary data when needed. With time and increased scale and variety of data the data processing got few more steps added. Currently data processing involves more steps which are collecting data, filtering it into meaningful data, sorting, processing which involves machine learning or similar algorithms, analyzing, storing, retrieving the necessary data and presenting it in an easy-to-understand format.

**Database** is an organized collection of data which are logically related. The most traditional and widely used database is Relational database. Recently more databases are becoming popular such as Graph database, Star schema, document database, key value pair database etc.

**Database Management System** is a software which acts as an interface between any database and the program or system using the database. The DBMS associated with relational database is called Relational Database Management System (RDBMS).

In this unit we are learning three types of databases which we will be implementing on a given use case. These are Relational, Graph and Document database where Relational is based on SQL and the other two are widely known as NoSQL databases.

# **Entity-Relationship Model**

Entity-relationship model is defined as a conceptual model which uses entities and relationship between these entities to visualize informational structure of problem or scenario.(IY. & P.P., 2009)**.** This is commonly used in database designing and complex system analysis. ERD are easier to conceptualize and can be converted to relational schema easily compared to directly creating relational model.

The main three elements of an ER diagram are:

|  |  |
| --- | --- |
| **Symbol** | **Element** |
| Entity | Entity: It represents things or beings  Or Strong Entity |
| Relationship | Relationship: Any relation between any two entities. |
|  | Attribute: The features of entities |

**TASK A.1: Choose a Use case and develop the ER model**

The use case we choose is use-case 2 where we will be creating database for academic publishing. We identified the following elements from the use case scenario provided:

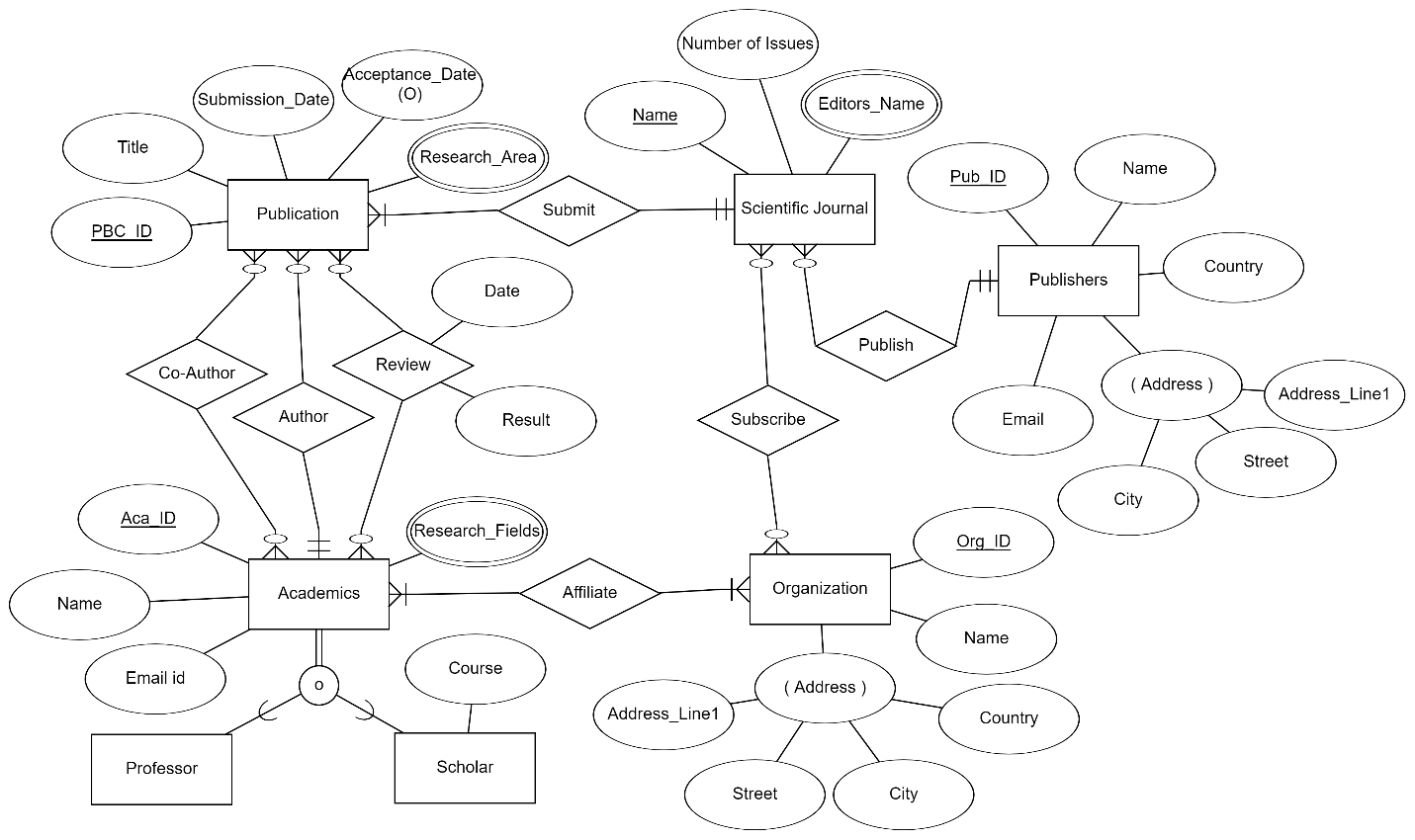
**Entities**: Publication, Scientific Journal, Publishers, Organization and Academics which we decided would be superclass for two subclasses Professor and Scholar.

**Relationships:** Publication is **Author**ed, **Co**-**Author**ed and **Review**ed by Academics, Publications are **Submit**ted to Scientific Journals, Academics are **Affiliat**ed with Organizations, Organizations **Subscribe** to Scientific Journals and Scientific Journals are **Publish**ed by Publishers.

**Attributes**: All the attributes were identified and added to respective entities, in which Publication’s Research Area, Academic’s Research Field and Scientific Journals Editors Name are multivalued attributes. Acceptance Date attribute of Publication is optional and Address are composite attributes.

**Tool Used**: erdplus.com

**ERD for Use case 2: Database for Academic Publication**

****

**RELATIONAL DATABASE**

**Relational database** is a series of interconnected tables where each table is a relation and the columns of the tables represents the features or attributes and each row called tuple represents a data. The tabular and relatable structure of relational database made it a popular choice and most companies used this database till early 20’s.

**Data Normalization** is the method of arranging the data in the most efficient form into a relational database to reduce redundant data and undesirable results such as insertion, deletion or updating anomaly. The data which is normalized is said to be in a Normal Form. We have the following Normal form in Relational Database:

* First Normal Form (1NF)
* Second Normal Form (2NF)
* Third Normal Form (3NF)
* Boyce-Codd Normal Form (BCNF)
* Fourth Normal Form (4NF)
* Fifth Normal Form (5NF)
* Sixth Normal Form (6NF)

Generally, while translating ERD to relational model will give us the schema already in the third normal form.

**TASK A.2 - Translating ER Diagram to Relational Model**

**(*Changes in each step are highlighted in blue*)**

**Step 1: Regular Entities**

**Method:** Add a table for each strong entity

* Regular Entities are Academics, Organization, Publication, Publishers and Scientific Journals.
* Academics – Aca\_ID is the primary key and Name and Email\_ID are other attributes for this table and we are not considering Research\_Fields here since it is a multi-valued attribute.
* Organization – Added Org\_ID as a Primary Key to make it easier. As Address is a Composite attribute, we have added all its attributes to the table and avoided address itself since its attributes combinedly make address.
* Publication – The Primary key for Publication is named as Pbc\_ID to avoid confusion with Publisher. We added all other attributes of Publication except Research\_Area as it is multivalued.
* Publishers – The Primary key here is Pub\_ID and the attribute is Name remaining all other attributes are multivalued.
* Scientific\_Journal – Name is considered as the Primary Key and No\_of\_issue is the only attribute and Editor’s name is not considered as it is multivalued.
* We have added the 5 tables for all the Regular Entities.

**Academics (****Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date**

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country**

**Scientific\_Journal (Name, No\_of\_issue**

**Step 2: Weak Entities**

**Method**: Add a table for each weak entity

* We don’t have any weak entities in our ER Diagram. So, we will skip this step.

**Step 3: One-to-one Relationships**

**Method**: Add Primary key of one Entity as Foreign Key in another Entity table and also add all attributes of relationship to same table

* We have don’t have any one-to-one relationship in our ER Diagram. So, we will skip this step as well.

**Step 4:** **One-to-many Relationships**

**Method**: Add the Primary key of one-sided entity to the table of the Many-sided entity as a Foreign key. Also add the simple attributes of the relationship to the same table.

* One-to-many Relationships are Author, Submit, and Publish.
* Author – This relationship is between Academics and Publication where Academics are the Author for Publication. Here, Author is the one-side and Publication is the Many-side as one Academic can be Author for more than one Publication but one Publication can have only one Author. So, we add the Primary key of Academic in the Publication table.
* Submit – This relationship is between Publication and Scientific\_Journal where Publications are Submitted to the Scientific\_Journals. Here, Publication is many-side and Scientific\_Journal is one-side as Publication can be submitted to only one Journal but a Journal can have many Publications. So, we add the Primary key of Scientific\_Journal in Publications table.
* Publish – This relationship is between Publisher and Scientific\_Journal where Publisher Publish the Scientific\_Journals. Here, Scientific\_Journal is many-side and Publisher is one-side as Publisher can be publish many Journals but a Journal can have only one Publisher. So, we add the Primary key of Publisher in Scientific\_Journal table.

**Academics (Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date, Aca\_ID\*, Journal\_Name\***

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country**

**Scientific\_Journal (Name, No\_of\_issue, Pub\_ID\***

**Step 5: Many-to-many Relationships**

**Method**: Add a new table for each many-to-many relationship and add the primary keys of both entities as foreign key who combinedly act as Primary key for this table

* Many-to-many Relationships are Co-author, Review, Affiliate and Subscribe.
* Co-author – This relationship is between Academics and Publication where Academics are the Co-author for Publication. An Academic can Co-author many Publications and a Publication can have more than one Co-authors.
* Review – This relationship is again between Academics and Publication, where Academics Review Publications. An Academic can review many Publications and a Publication can be reviewed by more than one Academic.
* Affiliate - This relationship is between Academics and Organization, where Academics are affiliated with Organizations. An Academic can affiliate with many Organizations and an Organization can be affiliated by many Academics.
* Subscribe - This relationship is between Scientific\_Journals and Organization, where Organizations are subscribed to Scientific\_Journals. An Organization can subscribe to many Journals and a Journal can be subscribed by many Organizations.

**Academics (Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date, Aca\_ID\*, Journal\_Name\***

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country**

**Scientific\_Journal (Name, No\_of\_issue, Pub\_ID\***

**Co-author ( Aca\_ID\*, Pbc\_ID\* )**

**Review ( Aca\_ID\*, Pbc\_ID\* , Date, Result)**

**Affiliate (Aca\_ID\*, Org\_ID\*)**

**Subscribe ( Org\_ID\*, Journal\_Name\*)**

**Step 6: Multi-valued Attributes**

**Method**: Add a new table for each multi-valued attribute with the Entities Primary key as Foreign key and the attribute.

* Multi-valued attributes are as below:
* Research\_Fields for Academics entity
* Research\_Area for Publication entity
* Editors\_Name for Scientific\_Journal entity

**Academics (Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date, Aca\_ID\*, Journal\_Name\***

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country**

**Scientific\_Journal (Name, No\_of\_issue, Pub\_ID\***

**Co-author ( Aca\_ID\*, Pbc\_ID\* )**

**Review ( Aca\_ID\*, Pbc\_ID\* , Date, Result)**

**Affiliate (Aca\_ID\*, Org\_ID\*)**

**Subscribe ( Org\_ID\*, Journal\_Name\*)**

**Research\_Fields (Aca\_ID\*, Research\_Field**

**Research\_Area (Pbc\_ID \*, Research\_Area**

**Editors\_Name (Journal\_Name\*, Editor\_Name**

**Step 7: N-ary Relationships**

**Method:** Add a new table for each n-ary with the primary keys of all entities as foreign keys and together forming the primary key of this table.

* We don’t have any n-ary relationship in our ERD.

**Step 8: Superclass and Subclass**

**Method**: Add a table for each sub class with the primary key of Superclass as Foreign key and other attributes.

* We have a Superclass Academic has two Subclasses.
* Professor and Scholar with Scholar having an attribute.
* We will add table for both subclass

**Academics (Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date, Aca\_ID\*, Journal\_Name\*)**

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country)**

**Scientific\_Journal (Name, No\_of\_issue, Pub\_ID\*)**

**Co-author ( Aca\_ID\*, Pbc\_ID\* )**

**Review ( Aca\_ID\*, Pbc\_ID\* , Date, Result)**

**Affiliate (Aca\_ID\*, Org\_ID\*)**

**Subscribe ( Org\_ID\*, Journal\_Name\*)**

**Research\_Fields (Aca\_ID\*, Research\_Field**

**Research\_Area (Pbc\_ID \*, Research\_Area**

**Editors\_Name (Journal\_Name\*, Editor\_Name**

**Professor (Aca\_ID\*,**

**Scholar (Aca\_ID\*,Course)**

**The final Relational Data Model Comparison**

**Academics (Aca\_ID , Name, Email\_ID**

**Organization (Org\_ID, Name, Address\_Line1, Street, City, Country**

**Publication (Pbc\_ID, Title, Submission\_Date, Acceptance\_Date, Aca\_ID\*, Journal\_Name\*)**

**Publisher (Pub\_ID, Name, Address\_Line1, Street, City, Country)**

**Scientific\_Journal (Name, No\_of\_issue, Pub\_ID\*)**

**Co-author ( Aca\_ID\*, Pbc\_ID\* )**

**Review ( Aca\_ID\*, Pbc\_ID\* , Date, Result)**

**Affiliate (Aca\_ID\*, Org\_ID\*)**

**Subscribe ( Org\_ID\*, Journal\_Name\*)**

**Research\_Fields (Aca\_ID\*, Research\_Field**

**Research\_Area (Pbc\_ID \*, Research\_Area**

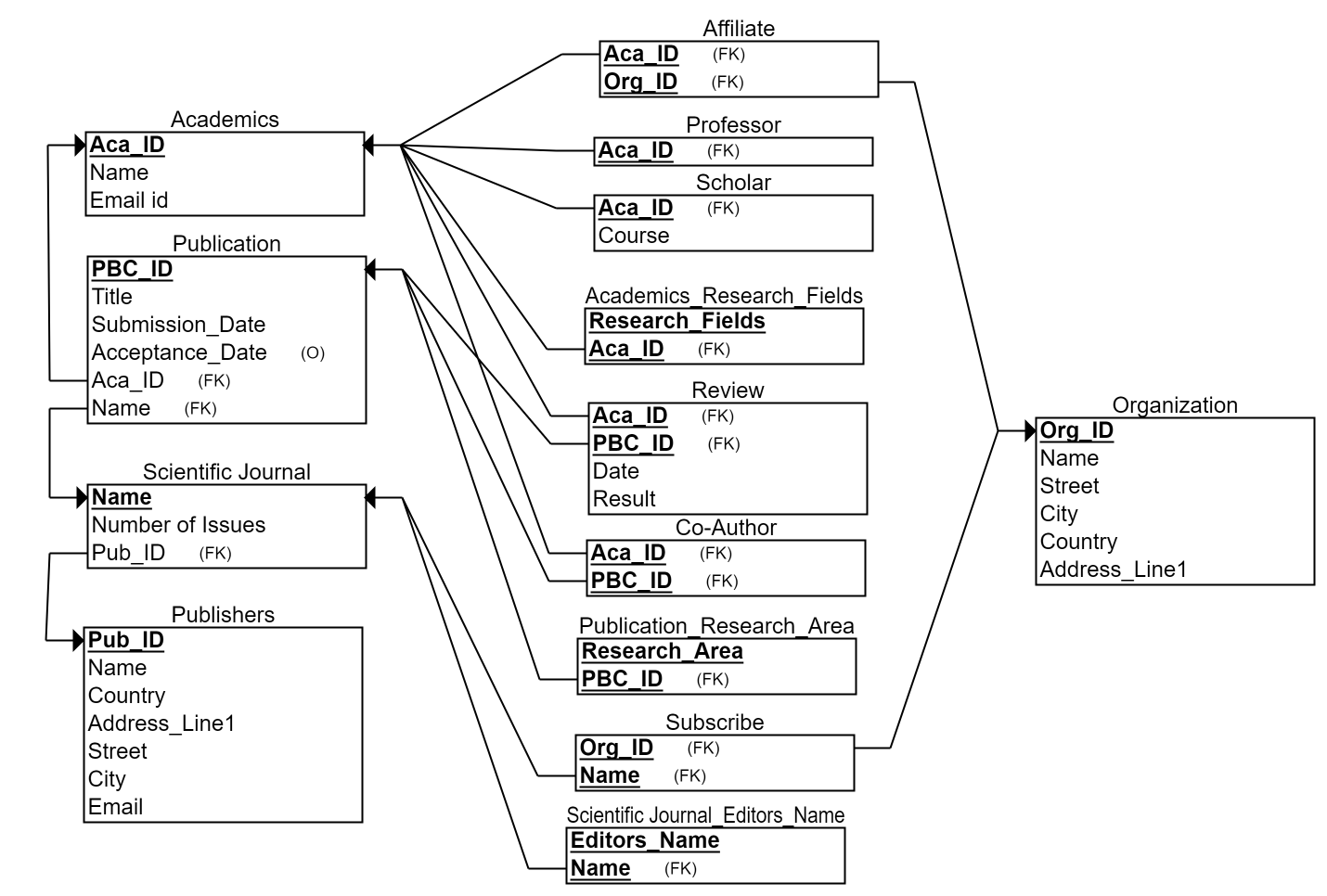
**Editors\_Name (Journal\_Name\*, Editor\_Name**

**Professor (Aca\_ID\*,**

**Scholar (Aca\_ID\*, Course)**

**Translation done through ERD Plus tool**

The Relational schema generated by the ERD tool is in accordance with our schema.



**Structured Query Language (SQL)**

SQL is a relational query language standardized to communicate with Relational databases and RDBMS. A single query using SQL is called as statements. There are different types of SQL statements, the two major types of statements are:

1. Data Definition Language (DDL): Used to design the database schema and manage the access to the data.
2. Data Manipulation Language (DML): Used to retrieve and update data.

Most used SQL statements are as below:

* CREATE database
* CREATE table
* INSERT INTO table VALUES data
* DELETE FROM table WHERE condition
* UPDATE data SET data WHERE condition
* SELECT data FROM table WHERE condition
* SELECT data FROM table GROUP BY Attribute
* SELECT data FROM table ORDER BY Attribute

**Advantages of SQL:**

* It is a non-procedural language
* SQL don’t throw indentation errors
* It is easy to learn due to high similarity of commands to English language
* Its portable and interactive with different viewpoint posibility

**Disadvantages of SQL:**

* Difficult to scale
* Not suitable for rapidly growing business
* Difficult to handle high volume data
* Cost Inefficiency

**Main Applications of SQL:**

* Write Data Integration Scripts
* Processing Analytical Queries
* Retrieving required information
* Transaction control such as insert, delete or update data.

**TASK A.3 - Implementing SQL, Reasoning Entity Choice & Test Cases**

We chose **Academic**, **Publication** and **Scientific Journal** for SQL Implementation.

**Major Reason**: SQL is efficient in handling complex queries and retrieve meaningful information by Joining tables. So, we chose the above since the Publication, Academics and Journal have multiple and using them to create a relational database can help get deeper insights.

***Create table for Academic Entity:***

**CREATE TABLE** academic**(**

aca\_id **NUMBER** (8,0),

name **VARCHAR2**(14),

email\_id **VARCHAR2** (30) not null,

**CONSTRAINT** pk\_academic **PRIMARY KEY (**aca\_id**)**

**);**

*Data Insertion Samples (5 insertion for Academic table)*:

**INSERT INTO** academic

**VALUES** (111, 'Anil K. Jain', 'aniljain@hmail.com');

**INSERT INTO** academic

**VALUES** (112, 'Salil Prabhakar', 'salprabh@hmail.com');

**INSERT INTO** academic

**VALUES** (113, 'Sharath Pankanti', 'sharathpan@hmail.com');

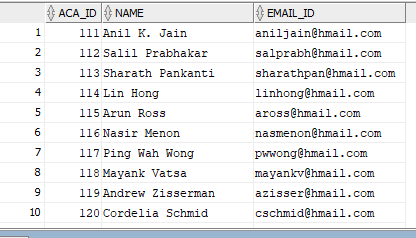
**INSERT INTO** academic

**VALUES** (114, 'Lin Hong', 'linhong@hmail.com');

**INSERT INTO** academic

**VALUES** (115, 'Arun Ross', 'aross@hmail.com');

**Table in SQL**



***Create table for Scientific\_Journal Entity:***

**CREATE TABLE** journal**(**

journal\_name  **VARCHAR2** (10) not null,

No\_of\_issue **NUMBER** (7,0)

**CONSTRAINT** pk\_journal **PRIMARY KEY (**journal\_name**)**

**);**

*Data Insertion Samples (5 insertion for journal table)*:

**INSERT INTO** journal

**VALUES** ('IEEE Transactions on Circuits and Systems for Video Technology', 2310, 101);

**INSERT INTO** journal

**VALUES** ('IEEE security & privacy', 3100, 101);

**INSERT INTO** journal

**VALUES** ('Pattern Recognition', 1100, 102);

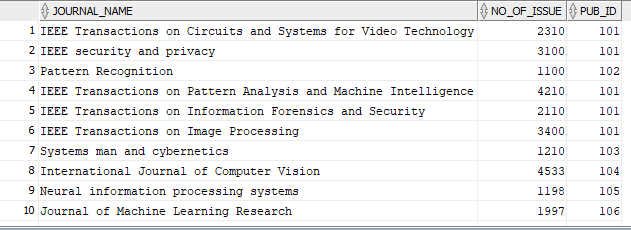
**INSERT INTO** journal

**VALUES** ('IEEE Transactions on Pattern Analysis and Machine Intelligence', 4210, 101);

**INSERT INTO** journal

**VALUES** ('IEEE Transactions on Information Forensics and Security', 2110, 101);

**Table in SQL**



***Create table for Publication Entity:***

**CREATE TABLE** publication**(**

pbc\_id **NUMBER** (8,0),

title  **VARCHAR2** (40),

submission\_date **DATE,**

acceptance\_date **DATE,**

aca\_id **NUMBER** (8,0),

journal\_name  **VARCHAR2** (10),

**CONSTRAINT** pk\_publication **PRIMARY KEY (**pbc\_id**),**

**CONSTRAINT** fk\_aca\_id **FOREIGN KEY (**aca\_id**) REFERENCES** academic**(**aca\_id**)**

**CONSTRAINT** fk\_ journal\_name **FOREIGN KEY (**journal\_name**)**

**REFERENCES** journal**(**journal\_name**)**

**);**

*Data Insertion Samples (5 insertion for publication table)*:

**INSERT INTO** publication

**VALUES** (1111, 'An introduction to biometric recognition', **TO\_DATE**('27-09-2003', 'dd-mm-yyyy'), **TO\_DATE**('01-01-2004', 'dd-mm-yyyy'), 111, 'IEEE Transactions on Circuits and Systems for Video Technology');

**INSERT INTO** publication

**VALUES** (1112, ‘Biometric recognition: Security and privacy concerns’, **TO\_DATE**('28-12-2002', 'dd-mm-yyyy'), **TO\_DATE**('03-04-2003', 'dd-mm-yyyy'), 112, 'IEEE security & privacy ');

**INSERT INTO** publication

**VALUES** (1113, 'On the similarity of identical twin fingerprints', **TO\_DATE**('30-10-2001', 'dd-mm-yyyy'), **TO\_DATE**('03-02-2002', 'dd-mm-yyyy'), 113 'Pattern Recognition');

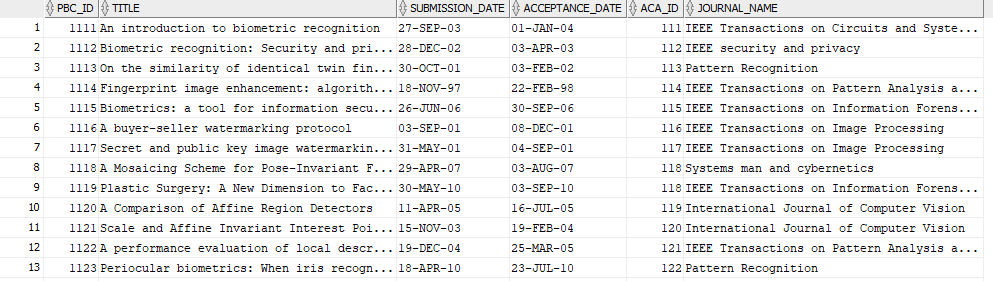
**INSERT INTO** publication

**VALUES** (1114, 'Fingerprint image enhancement: algorithm and performance evaluation', **TO\_DATE**('18-11-1997','dd-mm-yyyy'), **TO\_DATE**('22-2-1998','dd-mm-yyyy'), 114, 'IEEE Transactions on Pattern Analysis and Machine Intelligence');

**INSERT INTO** publication

**VALUES** (1115, 'Biometrics: a tool for information security', **TO\_DATE**('26-06-2006', 'dd-mm-yyyy'), **TO\_DATE**('30-09-2006', 'dd-mm-yyyy'), 115 'IEEE Transactions on Information Forensics and Security');

**Table in SQL:**



***Create table for Co-author Relation:***

**CREATE TABLE** coauthor**(**

aca\_id **NUMBER** (8,0),

pbc\_id **NUMBER** (8,0),

**CONSTRAINT** pk\_coauthor **PRIMARY key (**aca\_id, pbc\_id**),**

**CONSTRAINT** fk\_aca\_id\_co **FOREIGN KEY (**aca\_id**) REFERENCES** academic**(**aca\_id**)**

**CONSTRAINT** fk\_pbc\_id\_co **FOREIGN KEY (**pbc\_id**) REFERENCES** publication**(**pbc\_id**)**

**);**

*Data Insertion Samples (5 insertion for coauthor table)*:

**INSERT INTO** coauthor

**VALUES** (112, 1111);

**INSERT INTO** coauthor

**VALUES** (115, 1111);

**INSERT INTO** coauthor

**VALUES** (111, 1112);

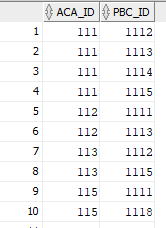
**INSERT INTO** coauthor

**VALUES** (113, 1112);

**INSERT INTO** coauthor

**VALUES** (111, 1113);

**Table in SQL:**



***Create table for Review Relation:***

**CREATE TABLE** review**(**

aca\_id **NUMBER** (8,0),

pbc\_id **NUMBER** (8,0),

date\_rv **DATE**,

result\_rv **VARCHAR2** (30),

**CONSTRAINT** pk\_review **PRIMARY key (**aca\_id, pbc\_id**),**

**CONSTRAINT** fk\_aca\_id\_rv **FOREIGN KEY (**aca\_id**) REFERENCES** academic**(**aca\_id**)**

**CONSTRAINT** fk\_pbc\_id\_rv **FOREIGN KEY (**pbc\_id**) REFERENCES** publication**(**pbc\_id**)**

**);**

*Data Insertion Samples (5 insertion for review table)*:

**INSERT INTO** review

**VALUES** (113, 1111, **TO\_DATE**('12-12-2003', 'dd-mm-yyyy'));

**INSERT** **INTO** review

**VALUES** (111, 1114, **TO\_DATE**('2-2-1998', 'dd-mm-yyyy'));

**INSERT INTO** review

**VALUES** (114, 1115, **TO\_DATE**('10-09-2006', 'dd-mm-yyyy'));

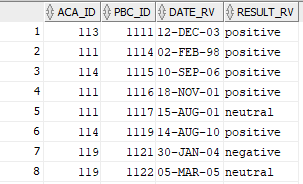
**INSERT INTO** review

**VALUES** (111, 1116, **TO\_DATE**('18-11-2001', 'dd-mm-yyyy'));

**INSERT INTO** review

**VALUES** (111, 1117, **TO\_DATE**('15-08-2001', 'dd-mm-yyyy'));

**Table in SQL:**



***Create table for Research Field attribute:***

**CREATE TABLE** research\_field**(**

aca\_id **NUMBER** (8,0),

research\_field **VARCHAR2** (30),

**CONSTRAINT** pk\_research\_field **PRIMARY key (**aca\_id, research\_field**),**

**CONSTRAINT** fk\_aca\_id\_rf **FOREIGN KEY (**aca\_id**) REFERENCES** academic**(**aca\_id**)**

**);**

*Data Insertion Samples (5 insertion for research field table)*:

**INSERT INTO** research\_field

**VALUES** (111, 'Artificial intelligence');

**INSERT INTO** research\_field

**VALUES** (111, 'Statistics');

**INSERT INTO** research\_field

**VALUES** (111, ' Computer vision');

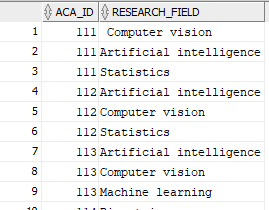
**INSERT INTO** research\_field

**VALUES** (112, 'Artificial intelligence');

**INSERT INTO** research\_field

**VALUES** (112, 'Statistics');

**Table in SQL:**



***Create table for Research Area attribute:***

**CREATE TABLE** research\_area**(**

pbc\_id **NUMBER** (8,0),

research\_area **VARCHAR2** (30),

**CONSTRAINT** pk\_research\_area **PRIMARY key (**pbc\_id, research\_area**),**

**CONSTRAINT** fk\_pbc\_id\_ra **FOREIGN KEY (**pbc\_id**) REFERENCES** publication**(**pbc\_id**)**

**);**

*Data Insertion Samples (5 insertion for research area table)*:

**INSERT INTO** research\_area

**VALUES** (1111, 'Computer vision');

**INSERT INTO** research\_area

**VALUES** (1112, 'Pattern Recognition');

**INSERT INTO** research\_area

**VALUES** (1113, 'Pattern Recognition');

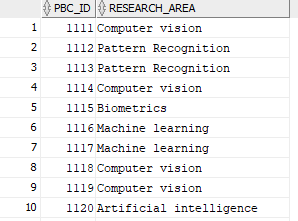
**INSERT INTO** research\_area

**VALUES** (1114, 'Computer vision');

**INSERT INTO** research\_area

**VALUES** (1115, 'Biometrics');

**Table in SQL:**



**SQL TEST CASES**

1. **Retrieve all Academic papers authored by ‘Nicholas R. Jennings’ with author details.**

**QUERY:**

**SELECT** \*

**FROM** publication p

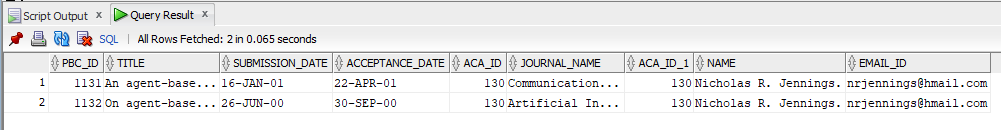
**JOIN** academic a **ON** p.aca\_id = a.aca\_id

**WHERE** p.aca\_id = (**SELECT** aca\_id

**FROM** academic

**WHERE** name = 'Nicholas R. Jennings.');

**OUTPUT:**



1. **Retrieve the academic papers name which are co-authored by ‘Anil K. Jain’.**

**QUERY:**

**SELECT** p.title, a.name **AS** COAUTHOR

**FROM** publication p

**JOIN** coauthor ca **ON** p.pbc\_id = ca.pbc\_id

**JOIN** academic a **ON** ca.aca\_id = a.aca\_id

**WHERE** p.pbc\_id IN (**SELECT** ca.pbc\_id

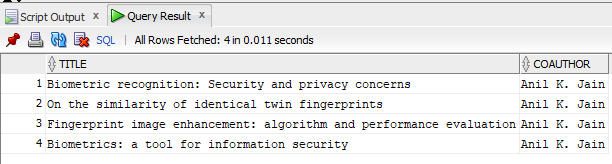
**FROM** coauthor

**WHERE** ca.aca\_id = (**SELECT** aca\_id

**FROM** academic

**WHERE** name = 'Anil K. Jain'));

**OUTPUT:**



1. **Retrieve the Journal in which ‘Arun Ross’ has published either as Author or Co-author**

**QUERY:**

**SELECT** j.journal\_name, a.name **AS** authorcoauthor

**FROM** journal j

**JOIN** publication p **ON** p.journal\_name = j.journal\_name

**JOIN** academic a **ON** p.aca\_id = a.aca\_id

**WHERE** p.aca\_id IN (**SELECT** aca\_id

**FROM** academic

**WHERE** name = 'Arun Ross')

**UNION**

**SELECT** j.journal\_name, a.name AS authorcoauthor

**FROM** journal j

**JOIN** publication p **ON** p.journal\_name = j.journal\_name

**JOIN** coauthor ca **ON** p.pbc\_id = ca.pbc\_id

**JOIN** academic a **ON** ca.aca\_id = a.aca\_id

**WHERE** p.pbc\_id IN ((**SELECT** ca.pbc\_id

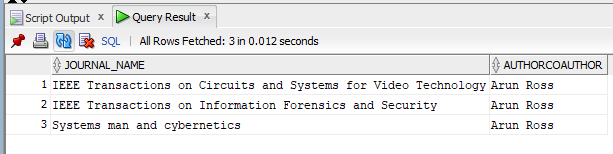
**FROM** coauthor

**WHERE** ca.aca\_id = (**SELECT** aca\_id

**FROM** academic

**WHERE** name = 'Arun Ross')));

**OUTPUT:**



1. **Retrieve the Academics names and other interested research fields whose Research Field is ‘The Internet’ in single row for each Academic.**

**QUERY:**

**SELECT** a.name, **LISTAGG** (rf.research\_field,' ,') **WITHIN** **GROUP**(order by rf.research\_field)

**FROM** academic a

**JOIN** research\_field rf **ON** a.aca\_id = rf.aca\_id

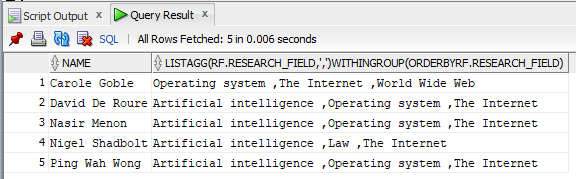
**WHERE** a.aca\_id **IN** (**SELECT** aca\_id

**FROM** research\_field

**WHERE** research\_field = 'The Internet')

**GROUP** **BY** a.name;

**OUTPUT:**



1. **Select the name and paper title of Academics whose Research\_Field is Computer Vision and published a paper with research area Computer vision.**

**QUERY:**

**SELECT** a.name, p.title

**FROM** academic a

**JOIN** research\_field rf **ON** rf.aca\_id = a.aca\_id

**JOIN** publication p **ON** p.aca\_id = a.aca\_id

**JOIN** research\_area ra **ON** ra.pbc\_id = p.pbc\_id

**WHERE** a.aca\_id IN ((**SELECT** rf.aca\_id

**FROM** research\_field

**WHERE** rf.research\_field = 'Computer vision')

**UNION**

(**SELECT** p.aca\_id

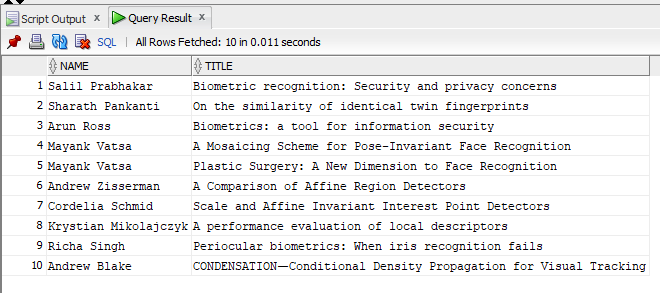
**FROM** publication

**WHERE** p.pbc\_id = (**SELECT** ra.pbc\_id

**FROM** research\_area

**WHERE** ra.research\_area = 'Computer Vision')));

**OUTPUT:**



**GRAPH DATABASE**

Graph database is a specific semantic schema designed to handle graph networks which consists of nodes, edges and properties. It can create, store, query and modify graph networks. It is also referred as semantic database due to its uniqueness for a specific type of structure. Graph database one of the widely used NoSQL database. Even though initially graph database was created to cater specific category of businesses like social media currently graph database support providers argue its applicability in various types of business.

Graph database is represented by the equation. Here G denotes a graph which consists:

**G = (V, E, LV, LE, ID)**

Where, V – set of nodes

E – set of edges

Lv – set of node labels

Le – set of edge labels

ID – Unique identifiers of nodes and edges

**Translating our ERD to Graph Definition**

**G = (V, E, LV, LE, ID)**

**LV = {****Academics,** **Publication, Journal, Publisher, Organization}**

**t Academics = {Academics, A Academics}**

**A Academics = {(Name, String), (Email\_id, String),(Research\_Field, String)}**

**t Publication = {Publication, A Publication}**

**A Publication = {(Title, String), (Submission\_Date, Integer), (Acceptance\_Date, Integer), (Research\_Area, String)}**

**t Journal = {Journal, A Journal}**

**A Journal = {(Name, String), (No\_of\_issue, Integer), (Editors\_Name, String)}**

**t Publisher = {Publisher, A Publisher}**

**A Publisher = {(Name, String), (Address\_Line1, String), (Street, String), (City, String), (Country, String)}**

**t Organization = {Organization, A Organization}**

**A Organization = {(Name, String), (Address\_Line1, String), (Street, String), (City, String), (Country, String)}**

**LE = {Author, Review, Coauthor, Submit, Subscribe, Affiliate, Publish}**

**t Author = {Author, ∅, {Academics}, {Publication}}**

**A Author = ∅**

**t Review = {Review, A Review, {Academics}, {Publication}}**

**A Review = {(Date), (Result)}**

**t Coauthor = {Coauthor, ∅, {Academics}, {Publication}}**

**A Coauthor = ∅**

**t Submit = {Submit, ∅, {Publication}, {Journal}}**

**A Submit = ∅**

**t Subscribe = {Subscribe, ∅, {Organization}, {Journal}}**

**A Subscribe = ∅**

**t Affiliate = {Affiliate, ∅, {Academics}, {Organization}}**

**A Affiliate = ∅**

**t Publish = {Publish, ∅, {Publisher}, {Journal}}**

**A Publish = ∅**

**Neo4J**

Network Exploration and Optimization 4 Java (Neo4J) is a graph database based on Java programming language which supports both graph data storage and processing. It is a native graph database and is highly scalable and robust. And is the leading graph database solution providers.

**Features of Neo4j**

* It has user friendly graph query language -Cypher, which resembles to SQL
* Supports multiple programming languages such as .Net, python, Java etc.
* Enormous resources and community support.

**Applications of Neo4j/graph database**

* Social Media analysis is the best use case for graph databases
* Recommendation engines in e-commerce
* Network management for Telecom and IT sectors

**Advantages of graph database**

* Easier to handle changing relationships
* These are extremely flexible and schema less

**Disadvantages of graph database**

* Not able to process high volume transactions
* Complicated and mass analytical queries are difficult to handle
* Does not provided MDM functionality
* Query latency increase with the volume of data the query will traverse

**TASK A.4 - Implementing Neo4j, Reasoning Entity Choice & Test Cases**

We chose Journal, Organization and Academic for Neo4j implementation

**Major Reason**: Graph database is useful when there is the formation of a graph between the entities here the relationship between Journal, Organization and Academic is many to many relationships. Also, the subscription and similar relationship might keep on changing with time in such situations graph database is useful.

***Creating Journal Nodes (Samples)***

**CREATE**(s1: Journal {Name:"IEEE Transactions on Circuits and Systems for Video Technology ", No\_of\_Issue : "2310", Editors:["Feng Wu"," Jiankun Hu"," Stefano Tubaro"]})

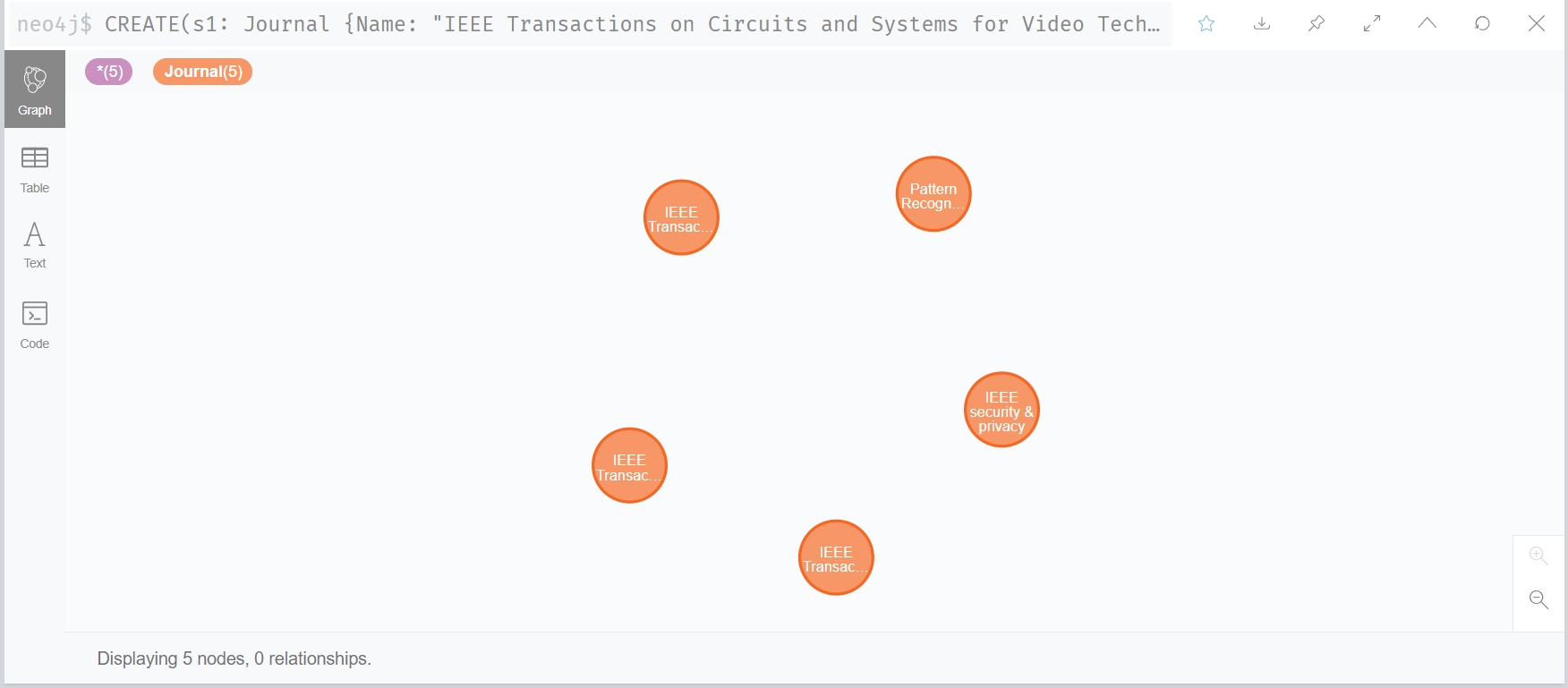
**CREATE**(s2: Journal { Name:"IEEE security & privacy ", No\_of\_Issue : "3100", Editors:["Sean Peisert"," Karthik Nandakumar"]})

**CREATE**(s3: Journal { Name:"Pattern Recognition ", No\_of\_Issue : "1100", Editors:["Edwin Hancock"," Ching Y. C. Y. Suen"]})

**CREATE**(s4: Journal { Name:"IEEE Transactions on Pattern Analysis and Machine Intelligence", No\_of\_Issue : "4210", Editors:["Kyoung Mu Lee"," Lejla Batina"]})

**CREATE**(s5: Journal { Name:"IEEE Transactions on Information Forensics and Security ", No\_of\_Issue : "2110", Editors:["Mauro Conti"," Chip Hong Chang"]})

**Sample Output:**



***Creating Organization Nodes (Samples)***

**CREATE**(o1: Organization{ Name:"Michigan State University", Address\_Line1:"426 Auditorium Road", Street:"East Lansing, MI 48824", City:"Michingan", Country:"USA" } )

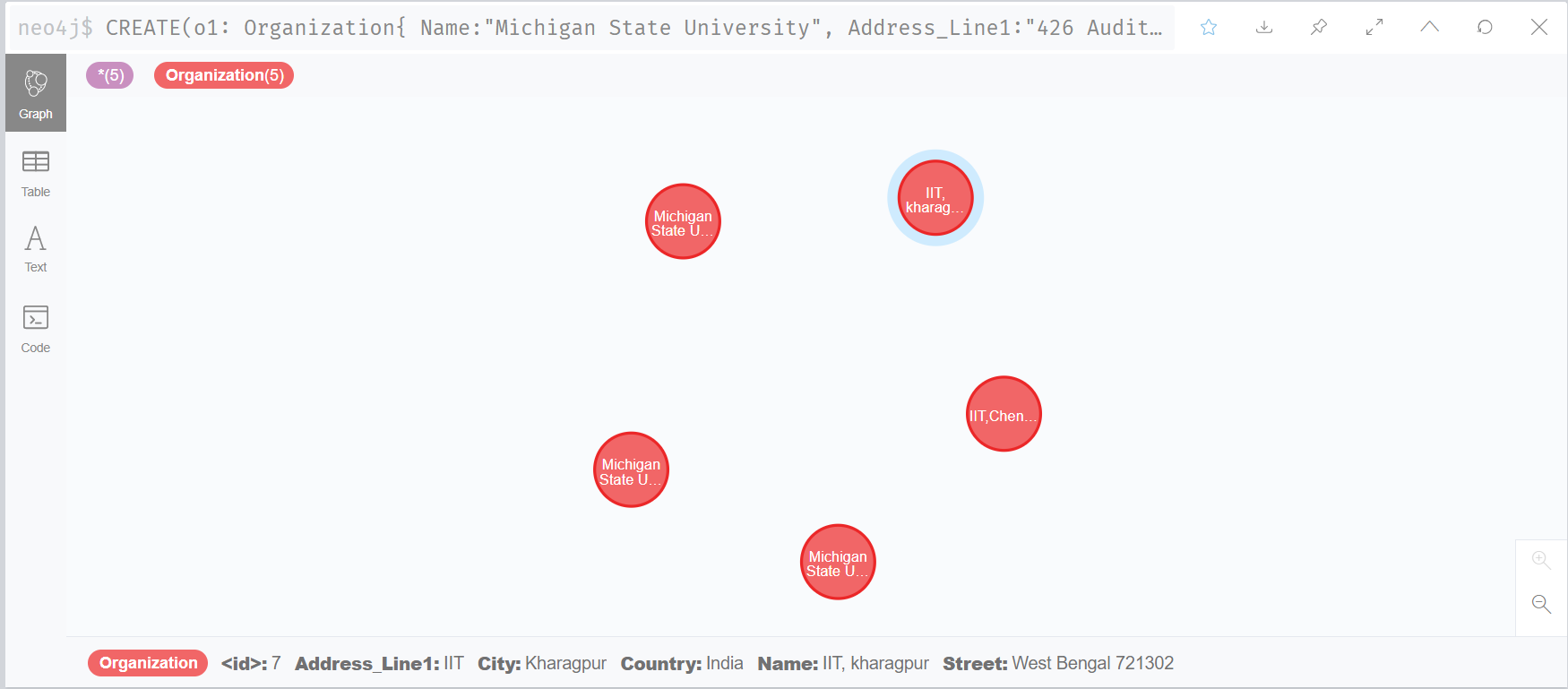
**CREATE**(o2: Organization{ Name:"IIT,Chennai", Address\_Line1:"IIT", Street:"Tamil Nadu 411233", City:"Chennai", Country:"India" } )

**CREATE**(o3: Organization{ Name:"IIT, kharagpur", Address\_Line1:"IIT", Street:"West Bengal 721302", City:"Kharagpur", Country:"India" } )

**CREATE**(o4: Organization{ Name:"Michigan State University", Address\_Line1:"426 Auditorium Road", Street:"East Lansing, MI 48824", City:"Michingan", Country:"USA" } )

**CREATE**(o5: Organization{ Name:"Michigan State University", Address\_Line1:"426 Auditorium Road", Street:"East Lansing, MI 48824", City:"Michingan", Country:"USA" } )

**Sample Output:**



***Creating Academic Nodes (Samples)***

**CREATE**(a1: Academic { Name: "Anil K. Jain", Email\_id: "anilkj@hmail.com", Research\_Field: ["Artificial intelligence", "Computer vision", "Statistics"]})

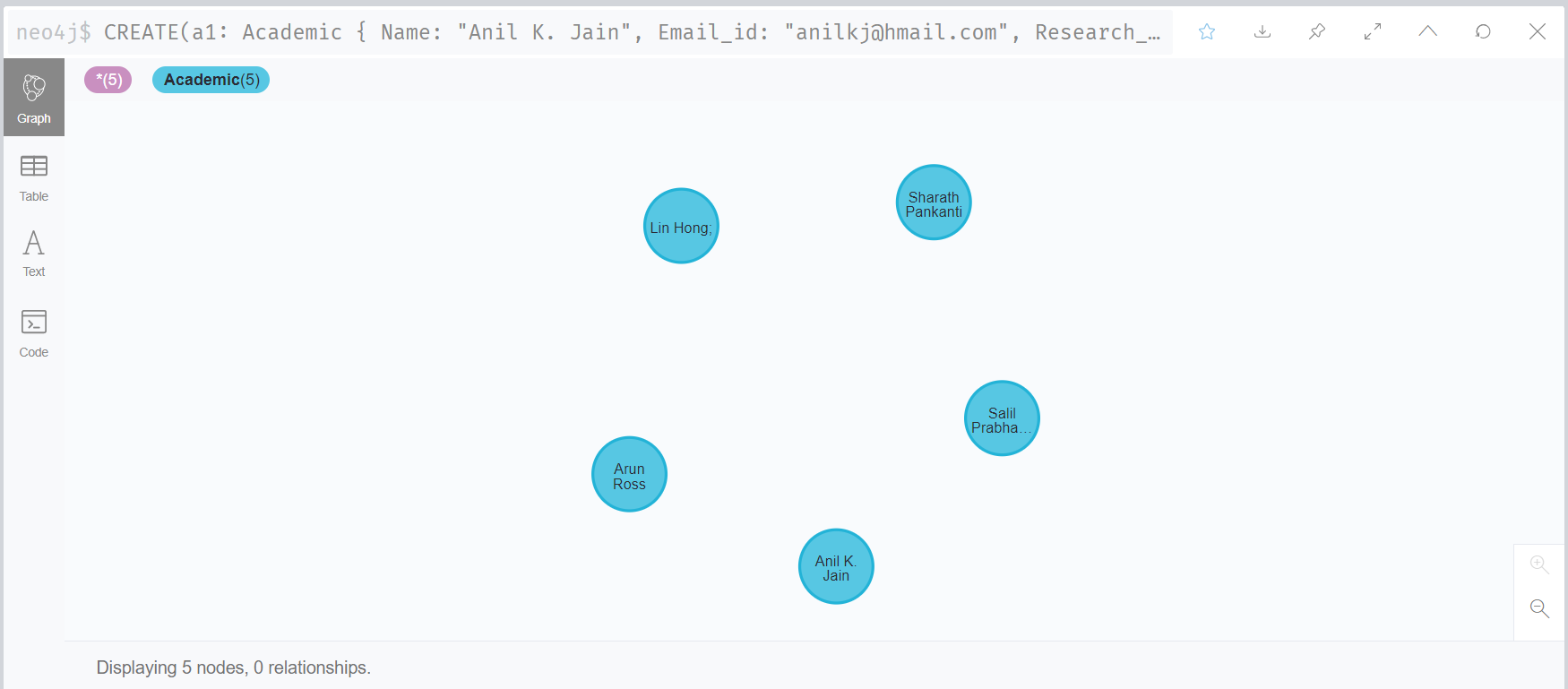
**CREATE**(a2: Academic { Name: "Salil Prabhakar", Email\_id: "sphrabha@hmail.com", Research\_Field: ["Artificial intelligence", "Computer vision", "Statistics"]})

**CREATE**(a3: Academic { Name: "Sharath Pankanti", Email\_id: "Sharathpan@hmail.com", Research\_Field: ["Artificial intelligence", "Computer vision", "Machine learning"]})

**CREATE**(a4: Academic { Name: "Lin Hong;", Email\_id: "linhong@hmail.com", Research\_Field: ["Biometrics", "Pattern Recognition"]})

**CREATE**(a5: Academic { Name: "Arun Ross", Email\_id: "aross@hmail.com", Research\_Field: ["Artificial intelligence","Computer vision","Machine learning"]})

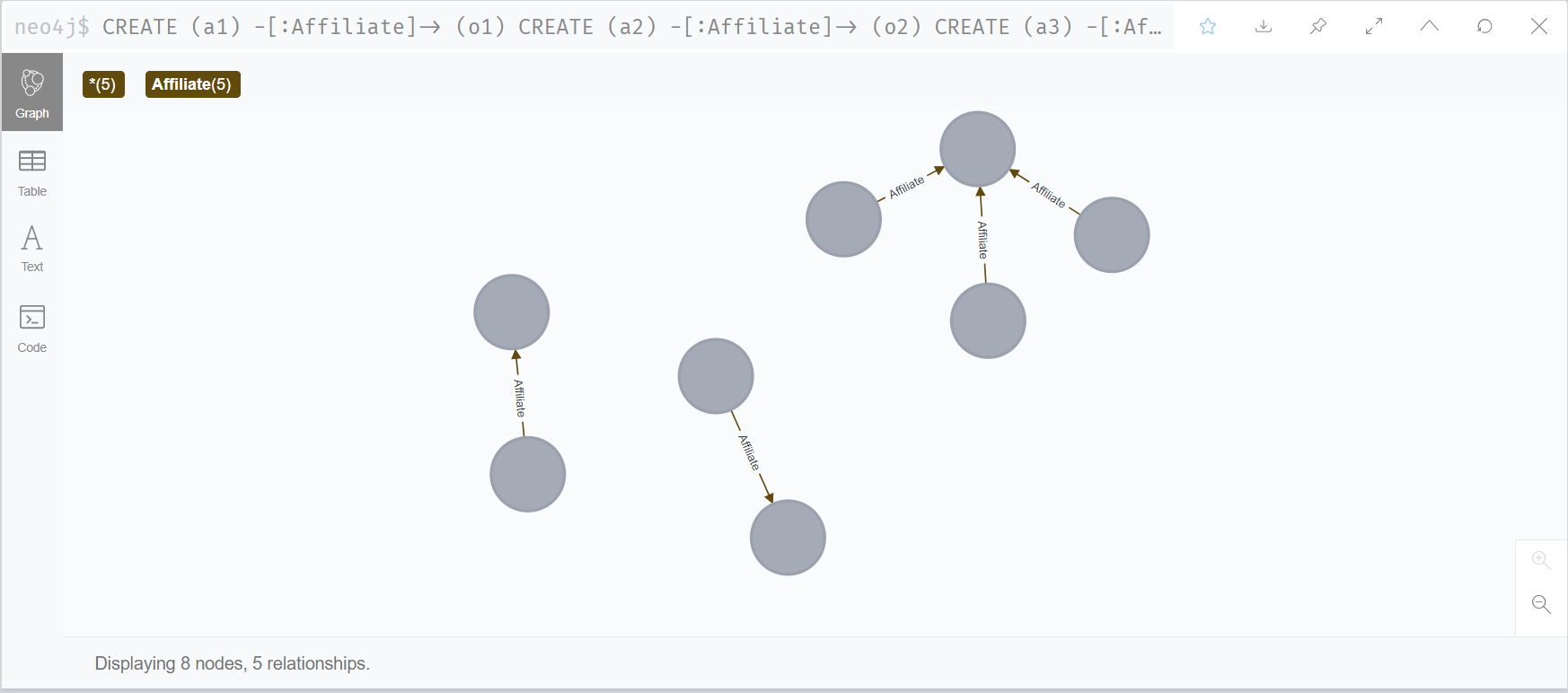
**Sample Output:**



***Creating Edge between Academic & Organization Nodes (Samples)***

|  |
| --- |
| **CREATE** (a1) -[:Affiliate]-> (o1) |
| **CREATE** (a2) -[:Affiliate]-> (o2) |
| **CREATE** (a3) -[:Affiliate]-> (o3) |
| **CREATE** (a4) -[:Affiliate]-> (o1) |
| **CREATE** (a5) -[:Affiliate]-> (o1) |

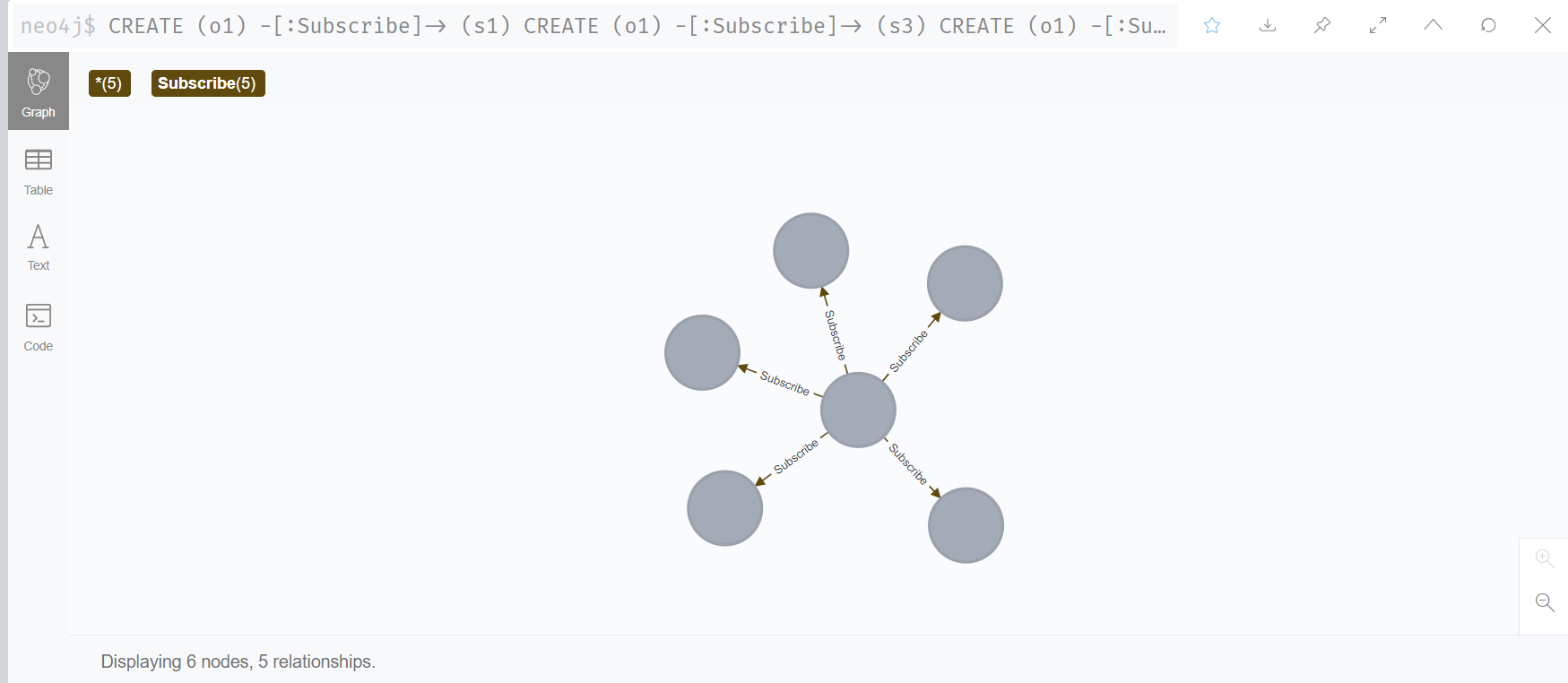
**Sample Output:**



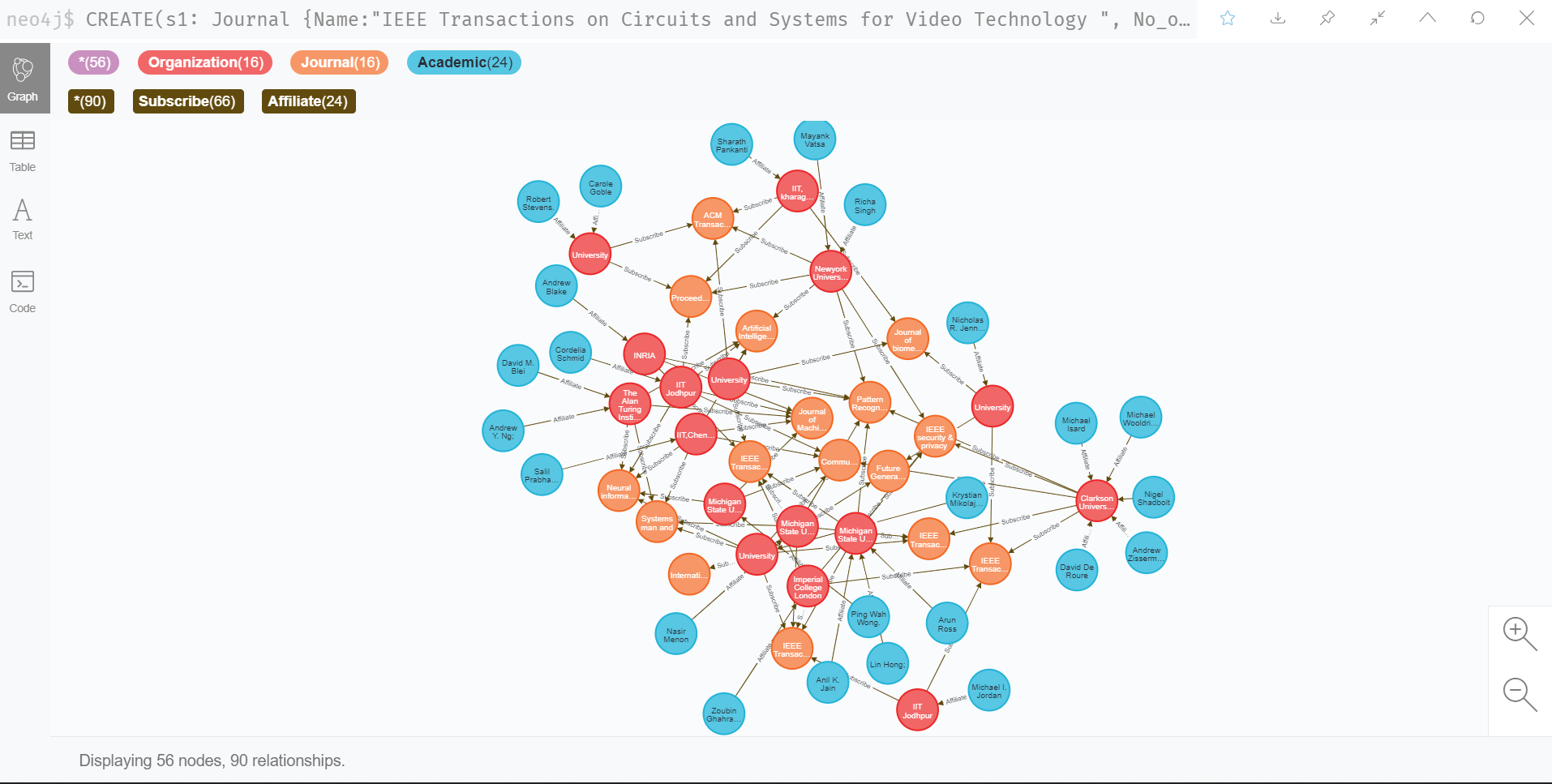
***Creating Edge between Journal & Organization Nodes (Samples)***

|  |
| --- |
| **CREATE** (o1) -[:Subscribe]-> (s1) |
| **CREATE** (o1) -[:Subscribe]-> (s3) |
| **CREATE** (o1) -[:Subscribe]-> (s2) |
| **CREATE** (o1) -[:Subscribe]-> (s5) |
| **CREATE** (o1) -[:Subscribe]-> (s6) |

**Sample Output:**



**Complete Graph Database created:**

****

**NEO4J TEST CASES**

1. **Which Academics are affiliated to Organization ‘Michigan State University’ and what are their Research field?**

**QUERY:**

**MATCH** (a:Academic{ })-[c: Affiliate]->( o:Organization{Name:"Michigan State University"})

**RETURN** a.Name,a. Research\_Field

**OUTPUT:**



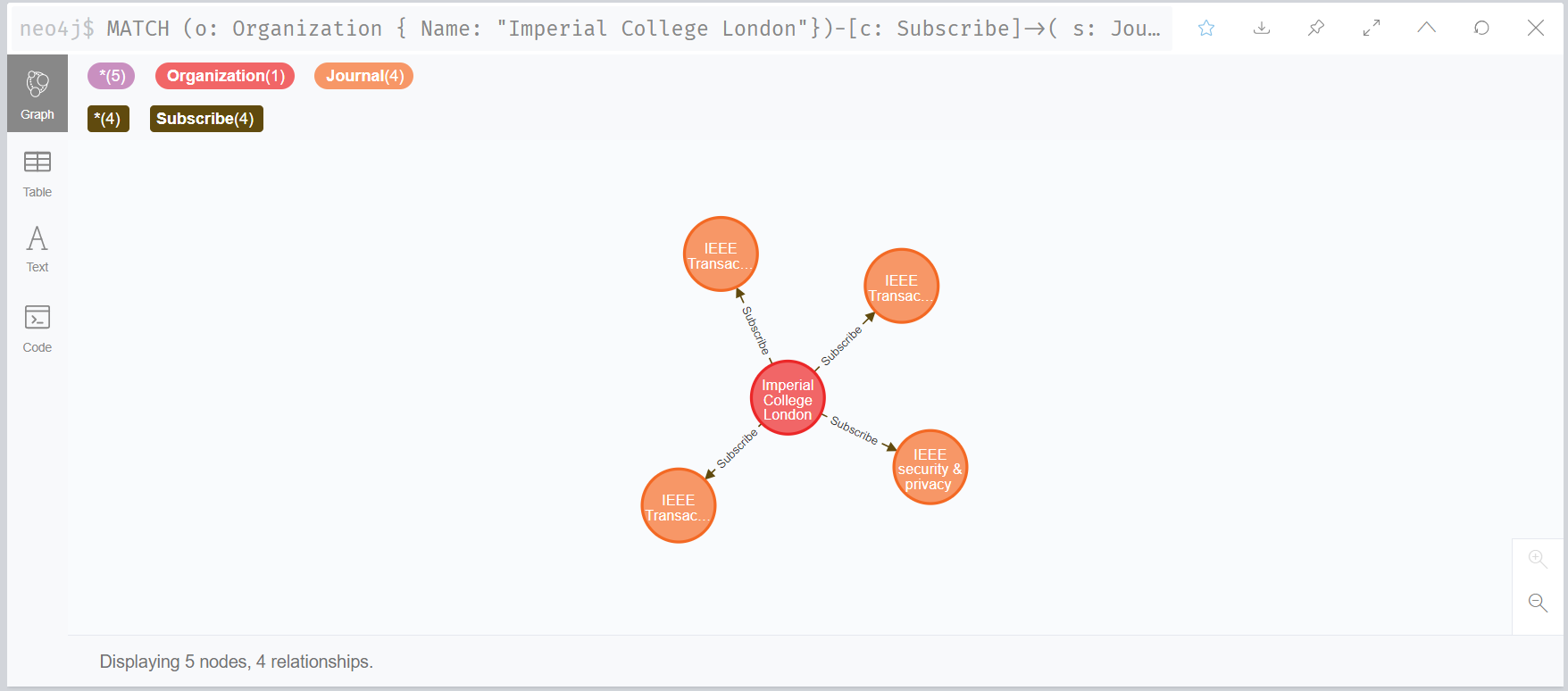
1. **Organization ‘Imperial College London’ is subscribed to which Journals?**

**QUERY:**

**MATCH** (o: Organization {Name: “Imperial College London”})-[c: Subscribe]->( s: Journal{ })

**RETURN** \*

**OUTPUT:**



1. **Get the Journals name subscribed by maximum number of universities and their names.**

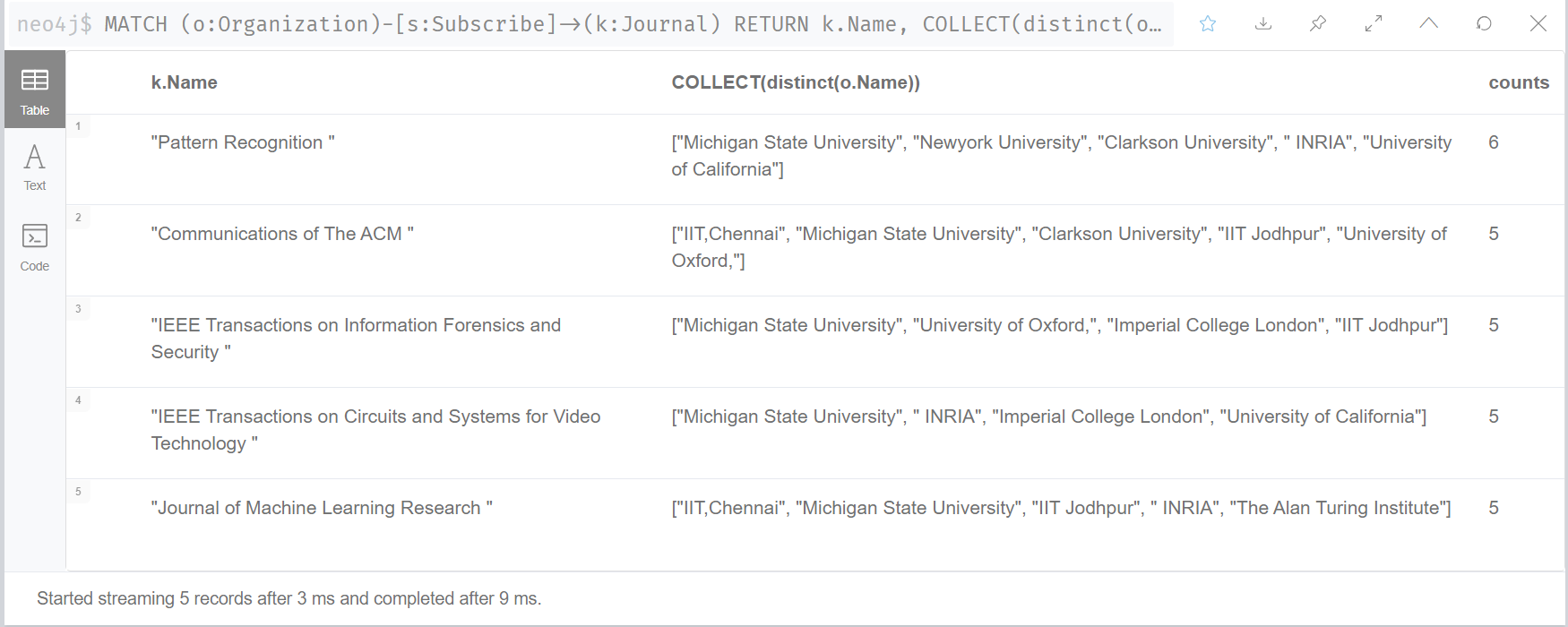
**QUERY:**

**MATCH** (o:Organization)-[s:Subscribe]->(k:Journal)

**RETURN** k.Name, **COLLECT**(distinct(o.Name)), **COUNT**(s) AS counts

**ORDER** **BY** counts **DESC**

**LIMIT 5OUTPUT:**

****

1. **Get Academics with both “Computer vision” and “Statistics” as their Research Field?**

**QUERY:**

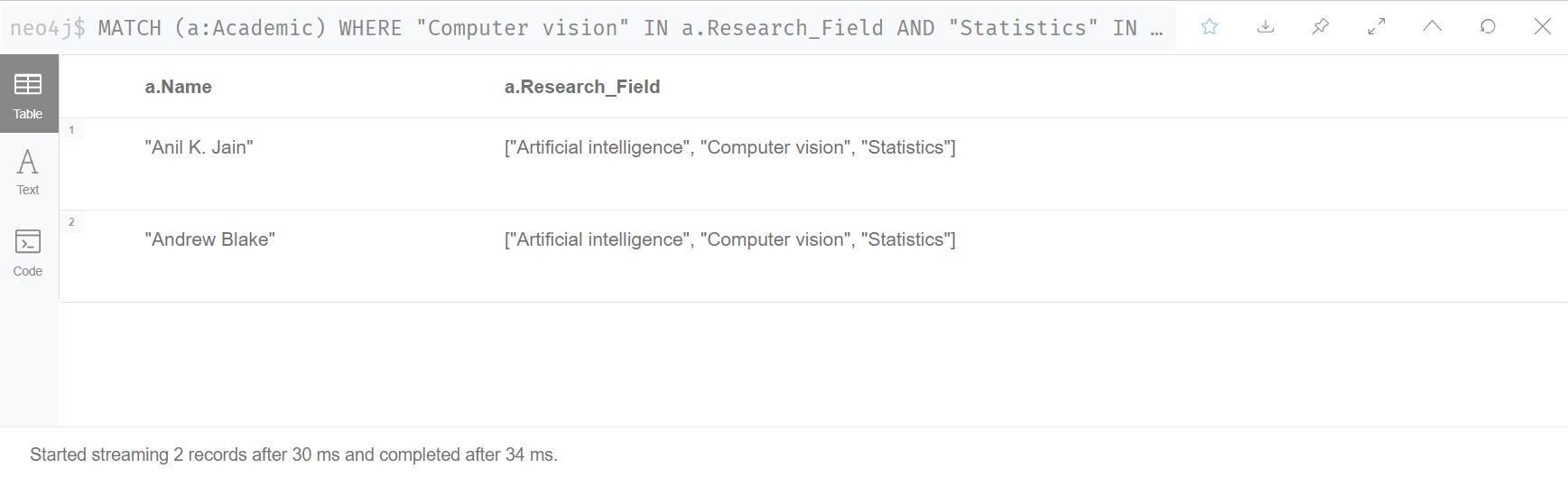
**MATCH** (a:Academic)

**WHERE** "Computer vision" **IN** a.Research\_Field

**AND** "Statistics" **IN** a.Research\_Field

**RETURN** a.Name, a.Research\_Field;

**OUTPUT:**

****

1. **Get Academics name affiliated to each Organization and show 5 organization with maximum number of academics affiliated?**

**QUERY:**

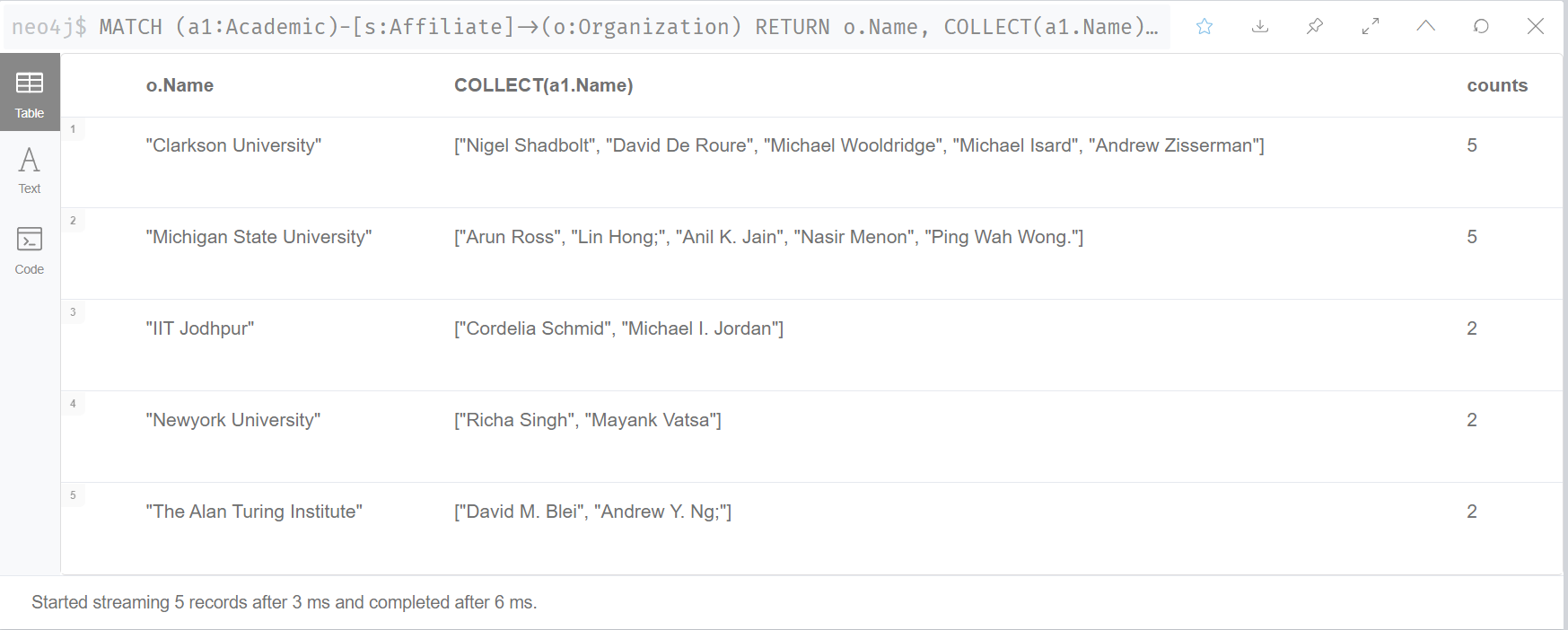
**MATCH** (a1:Academic)-[s:Affiliate]->(o:Organization)

**RETURN** o.Name, **COLLECT**(a1.Name), **COUNT**(s) AS counts

**ORDER** **BY** counts **DESC**

**LIMIT** 5

**OUTPUT:**

****

**DOCUMENT DATABASE**

Document database is another NoSQL database which has gained high popularity due to its structure less schema. It is a document-oriented database and stores data in the form of documents like JSON, CML, YAML, or binary documents. These documents are self-describing as it contains the data and the relations all in one document. Each document for similar entity can have different schema and also the schema can be changed by adding more details or removing at any time. Thus, it is useful when the entities vary with each other like a Superclass with several Subclasses. In document database we actually de-normalize the data to improve the speed.

**MongoDB**

MongoDB is an open-source document-based database program. In this data is stored in the form of JSON -like documents called BSON. Amazon has compared different Mongo DB features to SQL which gives us a better clarity of its structure.

|  |  |
| --- | --- |
| **Mongo DB Element** | **SQL Equivalent** |
| Collection | Table |
| Document (BSON) | Row |
| Field | Column |
| Index | Index |
| Embedded | Table Joins |

Mongo DB supports two types of relationships they are Embedding and Linking. These are also known as Data modelling.

**Embedding** is including the relationship data in that document itself. It helps to avoid joins and de-normalize data making it faster and easier to query.

**Linking** is also known as indexing or referencing as the name suggests it links a document to another document. But it nullifies the main purpose of document database which is denormalization and normalize the data.

Both relationships have its advantages and disadvantages and useful on its own in typical situations where the other won’t be appropriate.

**MongoDB Operations**

In MongoDB we can perform large number of operations which can be categorized into two:

* + - 1. **CRUD operations**

As the name suggests these operations deals with create, read, update and delete the documents in MongoDB Shell. (Other programming languages are also possible)

* + - * 1. **Create Operations**

To create collection

>db.createCollection(“database\_name”)

To insert documents in a collection

>db.Collection.insertOne([one document ])

> db.Collection.insertMany([ array of documents ])

* + - * 1. **Read Operations**

To find all document in a collection

>>db.Collection.find({})

To find documents with a condition in a collection

>db.Collection.find({ condition})

To find selected feature from documents with condition

>db.Collection.find({ condition},{Selected features:1})

* + - * 1. **Update Operations**

To update one document in a collection

>>db.Collection.updateOne(filter, update, options)

To update many documents in a collection

>db.Collection.updateMany(filter, update, options

To replace a document with another in a collection

>db.Collection.replace(filter, replacement, options)

* + - * 1. **Delete Operations**

To delete one document in a collection

>>db.Collection.deleteOne()

To delete many documents in a collection

>db.Collection.deleteMany()

* + - 1. **Aggregation Operations**

To approximate count documents in a collection

>db.collection.estimatedDocumentCount(options)

To count documents in a collection

> db.collection.count(query, options)

To distinct values for specified field in documents

> db.collection.distinct(field, query, options)

**Applications of MongoDB**

* Customer data management and personalization
* Agile software development
* Store product information by e-commerce and finance companies.

**Advantages of MongoDB**

* Document database is highly scalable and robust
* It is a schema less database giving flexibility with data
* It can handle variety of data structures making query and data distribution easier.
* Native edge to cloud data sync and is faster to build
* Support ad-hoc queries

**Disadvantages of MongoDB**

* Document databases require high memory
* Each document has a size limit and nesting capability
* Does not support transactions and joins

**TASK A.5 - Implementing MongoDB, Reasoning Entity Choice & Test Cases**

We chose Publication, Organization and Academic for Neo4j implementation

**Major Reason**: Document database is highly scalable and as the data for publication and academic keeps on scaling it's better to use it as document database. Academic and organizations also might have different subclasses and need different schema to store data document database support different schema for similar entities as well.

***Creating Academic Document with Embedded Organization entity (Samples)***

1. Create collection

**>** **db.createCollection(“**Academics**”)**

2. Insert Documents (samples shown only 2) – given as an array each document separated by comma

**> db.Academics.insertMany([**

**{**

"ID": "A01",

"Author": "Anil K. Jainn",

"Email": "anilkj@hmail.com",

"Research\_Field": ["Artificial intelligence", "Computer vision", "Statistics"],

"Organization": **{**

"Name": "Michigan State University",

"Address\_Line1": "426 Auditorium Road",

"Street": "East Lansing, MI 48824",

"City": "Michingan",

"Country": "USA"

**}**

**},**

**{**

"ID": "A02",

"Author": "Salil Prabhakar",

"Email": "sphrabha@hmail.com",

"Research\_Field": ["Artificial intelligence", "Computer vision", "Statistics"],

"Organization": **{**

"Name": "IIT,Chennai",

"Address\_Line1": "IIT ",

"Street": "Tamil Nadu 411233",

"City": "Chennai ",

"Country ": "India"

**}**

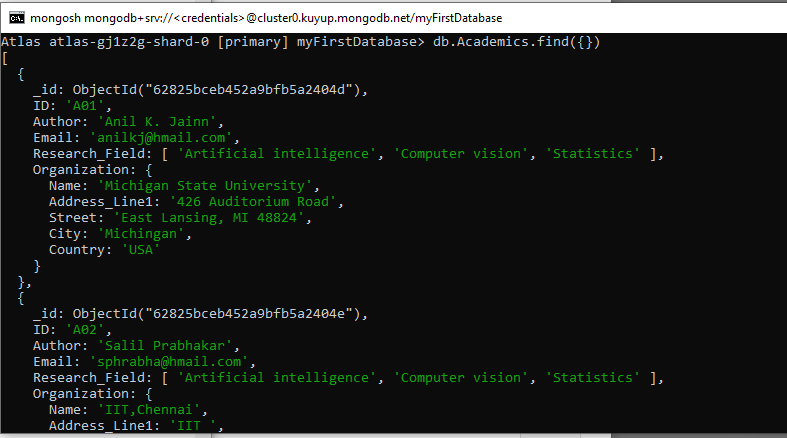
**}**

**])**

3. Check all the Documents in a collection

**> db.Academics.find( {} )**

**OUTPUT:**



***Creating Publication Document which is linked with Academic Document (Sample)***

While creating the Publication collection, we initially faced an issue with the Date category. Later after research we added the datatype in the JSON file as new Date for adding date. The format used is as below:

“Date\_varaible**”: new Date**(year, month, date)

We used this for both “Submission date” and “Approval date” variables in all data.

1. Create collection

**> db.createCollection(“**Publications**”)**

2. Insert Documents (samples only 2)- given as an array each document separated by comma

**> db.Publicaions.insertMany([**

**{**

"Title": "An introduction to biometric recognition",

"Research\_Area": "Computer vision",

"Author": "A01",

"CoAuthor": ["A05", "A02"],

"Submission\_Date": new Date (2003,9,27),

"Approval\_Date": new Date (2004,1,1)

**},**

**{**

"Title": "Biometric recognition: Security and privacy concerns",

"Research\_Area": "Pattern Recognition",

"Author": "A02",

"CoAuthor": ["A03", "A01"],

"Submission\_Date": new Date (2002,12,29),

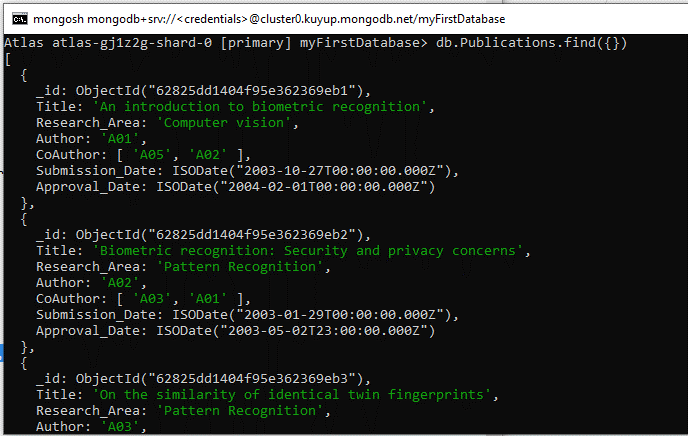
"Approval\_Date": new Date (2003,4,3)

**}])**

3. Check all the Documents in a collection

**> db.Publications.find( {} )**

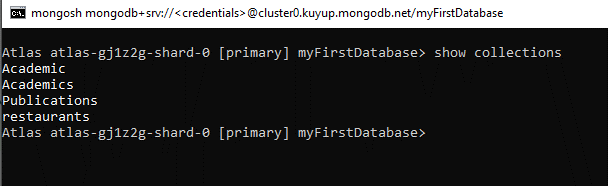
**OUTPUT:**



After Creating the collections, we used the below command to see the collections in our database:

**>show collections**

**OUTPUT:**



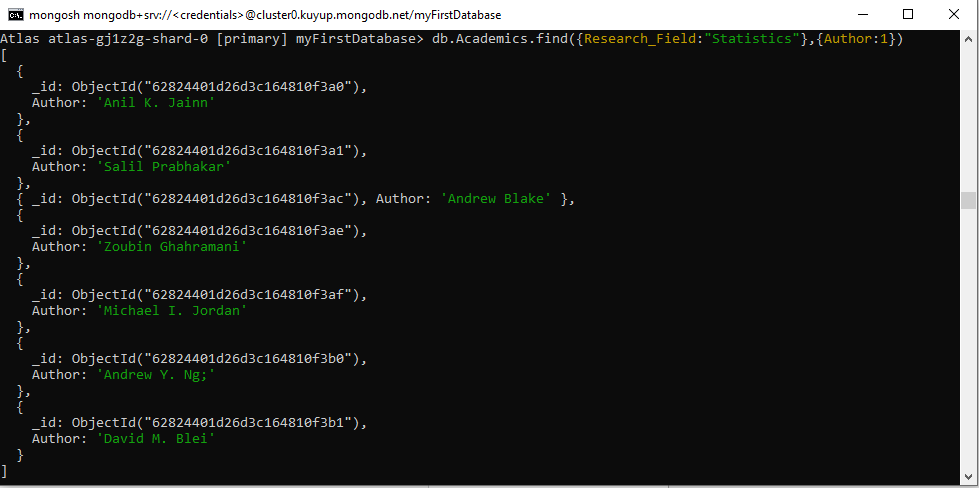
**MONGODB TEST CASES**

1. **Which Academics have ‘Statistics’ as their Research Field?**

**QUERY:**

**> db.Academics.find**( {$elemMatch: {" Research\_Field ": ‘The Internet}})

**OUTPUT:**

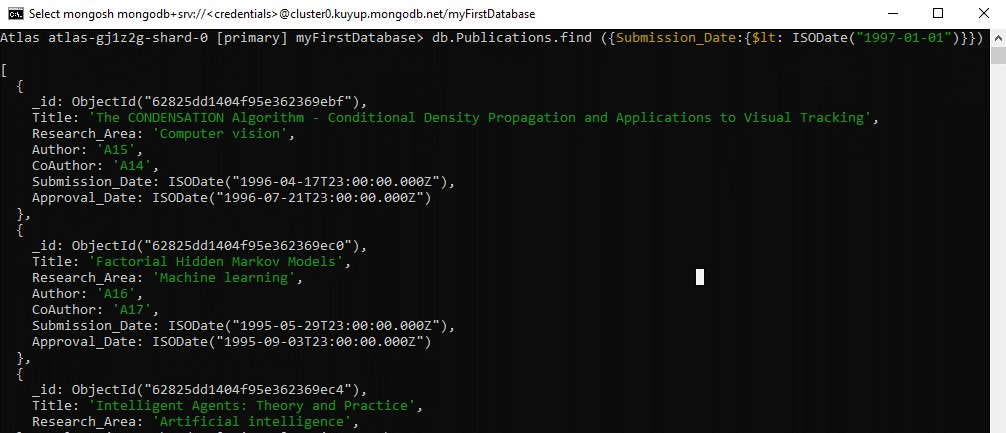


1. **Find publications that have submission date before the year 1997?**

**QUERY:**

**> db.Publication.find** ( {Submission\_Date: {$lt: ISODate ("1997-01-01")}})

**OUTPUT:**

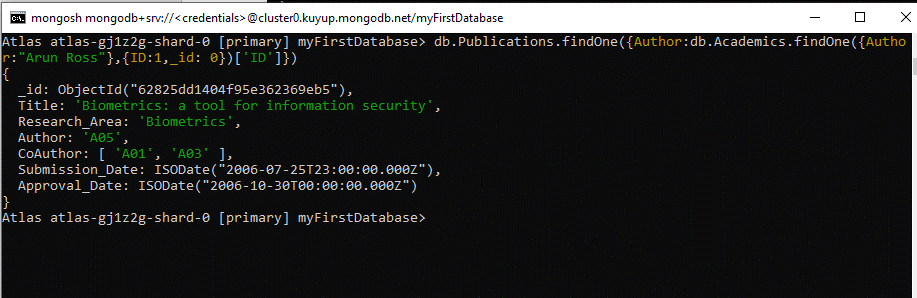


1. **Find a publication whose Author is "Arun Ross"?**

**QUERY:**

**> db.Publication.findOne**({Author: db.Academics.findOne({Author: "Arun Ross"},{ID: 1, \_id: 0}) ['ID']})

**OUTPUT:**

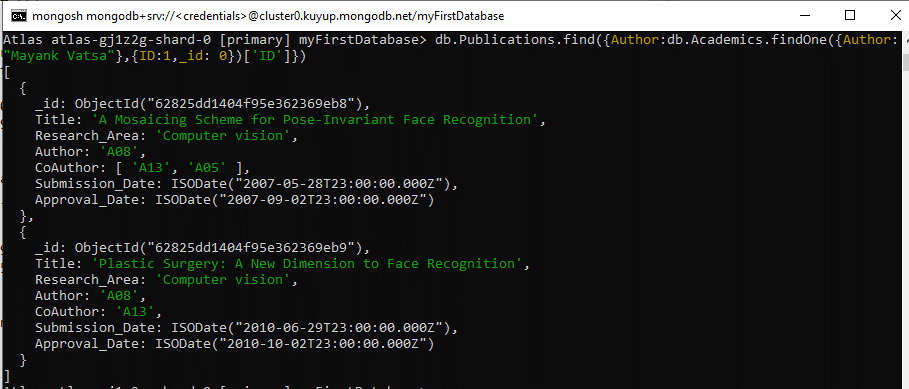


1. **Find all publication whose Author is "Mayank Vatsa"?**

**QUERY:**

**> db.Publication.find**({Author: db.Academics.findOne( {Author: "Mayank Vatsa"} , {ID: 1, \_id: 0}) ['ID']})]})

**OUTPUT:**



1. **Find a publication whose Author is "Carole Goble" and Co-Author is Robert Stevens?**

**QUERY:**

**> db.Publication.find** ({Author: db.Academics.findOne({Author: "Carole Goble"},{ID: 1, \_id: 0}) ['ID'],CoAuthor: db.Academics.findOne( {Author: "Robert Stevens."}, {ID: 1, \_id: 0}) ['ID']})

**OUTPUT:**

