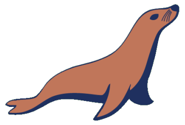
Topic 2
Introduction to Databases

Dr Diarmuid Ó Briain



Licence

Module Objectives

At the end of this topic the learner will:

- Distinguish Data Modelling Tools
- Write simple File-Based Data Models
- Classify different database types
- List the 12 Rules of Relational Databases
- Discuss Fundamental Database Concepts - ACID
- List databases available on the market.

Data Modelling Tools

- Create, Edit, and View Data Models
  - Editing data entities, relationships, and attributes
- Reverse Engineering
- Code Generation
  - Data modelling tools to generate Structured Query Language (SQL)
- Collaboration and Sharing
  - Facilitate teams to work on the same models simultaneously
- Documentation and Reporting

Benefits of Using Data Modelling Tools

- Data is structured correctly and relationships are maintained, leading to improved data quality and integrity
- Optimise database performance, enabling efficient data storage, retrieval, and analysis
- Automate many tasks, saving time and effort in the data modelling process, ultimately reducing overall development costs
- Better communication among stakeholders, promoting a shared understanding of the data landscape

TUS

File-Based Data Models

TUS

File Based Data Models

- Data is stored in separate files, each with its own structure
- Used for small to medium-sized applications where the data volume is relatively low
- Simple to implement and manage, but inefficient as the data volume increases
- Key Characteristics of File-Based Data Models
  - Data Separation
  - Modular Approach
  - Direct File Access
  - Limited Metadata Management
- Disadvantages
  - Inefficient with performance degrading as data volume increases
  - No centralised data management and indexing
  - Data duplication can easily occur
  - Limited data analysis

TUS

File Based Data Models

- File-based data model example storing data about orders, customers, and products
- This approach is not as efficient as a relational database for larger applications with more complex data relationships

```
~$ cat orders.txt
Order ID|Customer ID|Product ID|Quantity|Price
---------|------------|------------|--------|--------
1|1234|5678|10|10.00
2|4321|7654|20|25.00
3|9876|3456|30|32.50
```

```
~$ cat customers.txt
Customer ID|Name|Address|Phone Number
------------|--------|------------|---------
1|John Ryan|3 Mulgrave Street|087-456-7890
2|Tomas Smith|21 Sarsfield Street|086-678-9012
3|Peter Gleeson|35 Main Street|087-890-1234
```

```
~$ cat products.txt
Product ID|Name|Description|Price
---------|--------|------------|--------
1|Laptop|A powerful laptop for work and play.|1000.00
2|Smartphone|A high-end smartphone with a great camera.|500.00
3|Television|A 4K Ultra HD TV with HDR. |1200.00
```

TUS

What Is a Database?

- An organised collection of data, an electronic system that allows data to be easily accessed, manipulated and updated
- Provide a structured and controlled way to store, manage, and retrieve information, enabling organisations to harness the power of data for improved decision-making and innovation
- Databases are strategic business tools that empower organisations to extract meaningful insights and gain competitive advantages
- With databases, businesses can optimise operations, identify market trends, and drive growth

TUS

History of Databases – Great library at Alexandria – 260BC

History of Databases

- Databases have been around for centuries, from the ancient Egyptians to the Library of Alexandria (Ptolemy II Philadelphus ─ 285–246 BC)
- The computerisation of databases in the 1960s has revolutionised their capabilities
- Early databases were navigational, relying on pointers to navigate through data
- These were inefficient and limited in their ability to handle complex data relationships
- The relational model provides a more structured and efficient way to store and retrieve data
- Relational databases are now the standard for modern applications
- They are powerful, efficient, and easy to use

TUS

History of Database Systems

- Edgar Codd's relational model, published in 1970, was a new approach to storing and retrieving data
- The relational model organised data into tables with rows and columns

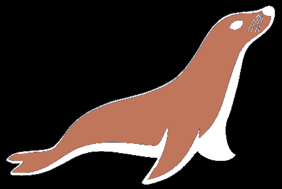
Edgar Codd

Larry Ellison

- Much easier to query and manipulate data, and it paved the way for the development of relational databases
- Codd's work was initially met with resistance from IBM, which invested in its own hierarchical DB model, Information Management System (IMS)
- However, Codd's vision eventually prevailed, and relational databases became the standard for commercial applications
- In 1979, Larry Ellison founded Oracle Corporation and adopted Codd's relational model for his database product

TUS

Relational Databases



TUS

Relational Database

- A relational database is a collection of tables that store data in a structured and organised way
- Each table has rows and columns, and the tables are related to each other by foreign keys
- **Referential integrity** is a rule that ensures that the relationships between tables are always consistent
  - This means that any changes to data in one table must also be reflected in the tables that relate to it
  - If a customer is deleted from the `CUSTOMER_MASTER` table, then that customers accounts must also be deleted from the `ACCOUNTS_MASTER` table
  - Referential integrity is important because it helps to prevent data from becoming corrupt
  - It also makes it easier to query and manipulate data

TUS

Relational Database

	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

TUS

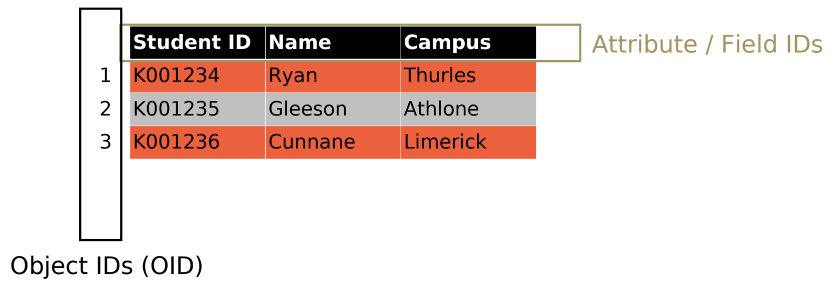
Relational Database

	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

Attribute / Field IDs

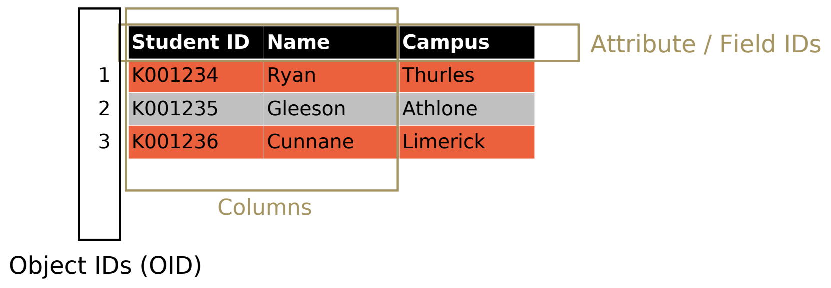
TUS

Relational Database



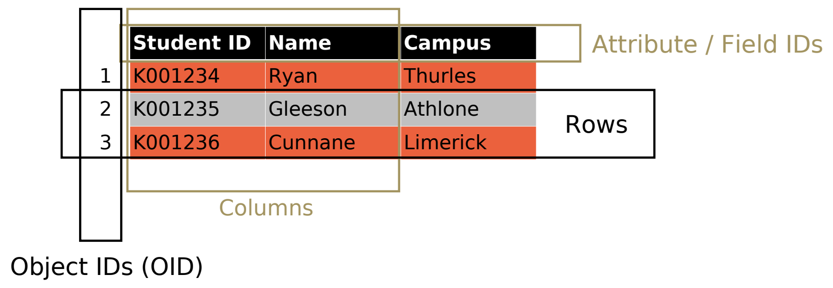
	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

Relational Database



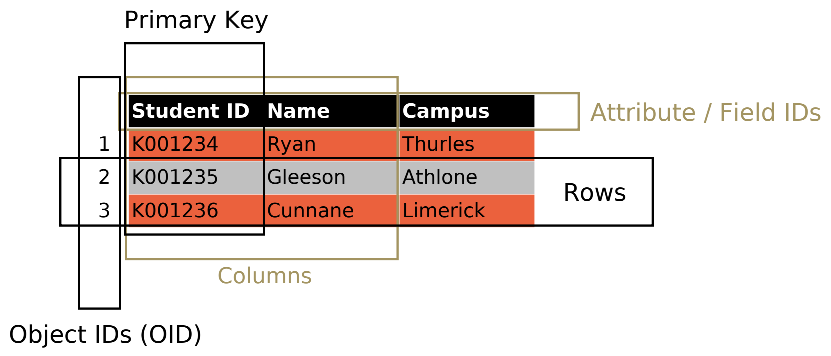
	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

Relational Database



	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

Relational Database



	Student ID	Name	Campus
1	K001234	Ryan	Thurles
2	K001235	Gleeson	Athlone
3	K001236	Cunnane	Limerick

Example DB tables

Orders table

Column Name	Data Type	Description
OrderID	INT	Primary key, Unique ID for each order
CustomerID	INT	Foreign key referencing the CustomerID in the Customers table
ProductID	INT	Foreign key referencing the ProductID in the Products table
Quantity	INT	Quantity of the product ordered
Price	FLOAT	Price of the product

Customers table

Column Name	Data Type	Description
CustomerID	INT	Primary key, Unique ID for each customer
Name	VARCHAR(50)	Customer's name
Address	VARCHAR(255)	Customer's address
PhoneNumber	VARCHAR(20)	Customer's phone number

Column Name	Data Type	Description
ProductID	INT	Primary key, Unique ID for each product
Name	VARCHAR(50)	Product’s name
Description	VARCHAR(255)	Product’s description
Price	FLOAT	Price of Product

Products table

Select data from DB tables

```
SQL> select * from orders
| orderID | customerID | productID | orderDate | status |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 2023-10-04 | pending |
| 2 | 2 | 2 | 2023-10-05 | processing |
| 3 | 3 | 3 | 2023-10-06 | delivered |

SQL> select * from customers
| customerID | customerName | address | email | phoneNo |
| --- | --- | --- | --- | --- |
| 1 | John Ryan | 3 Mulgrave St | john@gmail.com | 087-456-7890 |
| 2 | Thomas Smith | 21 Sarsfield St | tom@micro.org | 086-678-9012 |
| 3 | Peter Gleeson | 35 Main St | peter@itservice.com | 087-890-1234 |

SQL> select * from products
| productID | productName | price | description |
| --- | --- | --- | --- |
| 1 | Laptop | €1,000 | A powerful laptop for work and play. |
| 2 | Phone | €500 | A high-end smartphone with a great camera. |
| 3 | TV | €1200 | A 4K Ultra HD TV with HDR. |
```

Other Databases

Spreadsheets

- Simple data structures, not designed for multi-user access, and this can lead to conflicts and data corruption
- Spreadsheets don’t enforcing data validation and integrity rules
- Spreadsheets not good at running complex queries, join multiple tables, and create advanced reports
- Spreadsheets are not scalable, struggle to handle data volumes
- Spreadsheets are easier to use than databases; however, databases are more powerful and efficient for large-scale data management

Non relational Databases (NoSQL)

- Does not store data in a traditional relational format
- Store data in other ways, such as:
  - key-value pairs
  - Documents
  - Graphs
- More flexible and scalable than relational databases, which can be a good thing for certain types of applications.

Example: MongoDB

- Cross-platform, document-oriented database management system (DBMS) that stores data in JavaScriptObject Notation (JSON)-like documents
- NoSQL database that does not adhere to the strict schema requirements of traditional relational databases
- Stores data in flexible, self-describing JSON documents, enabling efficient storage and retrieval of complex data structures

Example: MongoDB

```
~$ cat mongodb_example.py
#! /usr/bin/env python3

import pymongo

# Connect to the MongoDB database
client = pymongo.MongoClient()
db = client['mydatabase']

# Get the products collection
products = db['products']

# Find all products with a price greater than 100
products_with_high_price = products.find({'price':{'$gt':100}})

for product in products_with_high_price:
    print(product)

# Find all orders that are pending
orders_pending = db['orders'].find({'status':'pending'})

for order in orders_pending:
    print(order)

```

```
~$ ./mongodb_example.py
{'_id': 1, 'productName': 'Laptop', 'price': 1000,
'description': 'A powerful laptop for work and play.'}
{'_id': 3, 'productName': 'TV', 'price': 700,
'description': 'A 4K Ultra HD TV with HDR.'}

{'_id':29,'customerID':3,'productID':3,'orderDate':2023-
10-06,'status':'pending'}

{'_id':30,'customerID':2,'productID':2,'orderDate':2023-
10-05,'status':'pending'}

{'_id':31,'customerID':1,'productID':1,'orderDate':2023-
10-04,'status':'pending'}
```

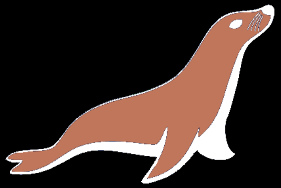
Relational vs Non relational Databases

Feature	Relational database	Non-relational database
Data storage	Tables	Key-value pairs, documents, graphs
Data modelling	Stricter	More flexible
Scalability	Less scalable	More scalable
Use cases	Enterprise applications, transactional systems	Web-based applications, cloud computing, social networking

Summary

	Spreadsheets	Non-relational databases	Relational databases	Data warehouses
Data storage	Flat files	Key-value pairs	Tables	Tables
Data integrity	Limited	Weaker	Stronger	Stronger
Scalability	Not scalable	Highly scalable	Less scalable	Highly scalable
Use cases	Personal data management, basic calculations	Web-based applications, cloud computing, social networking	Enterprise applications, transactional systems	Business intelligence, data analysis, trend forecasting

12 Rules of Relational Databases



Rules of Relational Databases

- Rule 0: **Foundation Rule**
  - "*A RDBMS must manage its stored data using only its relational capabilities*"

Rules of Relational Databases

- Rule 0: **Foundation Rule**
  - "*A RDBMS must manage its stored data using only its relational capabilities*"
- Rule 1: **Information Rule**
  - "*All information in the database should be represented in one and only one way – as values in a table*"

Rules of Relational Databases

- Rule 0: **Foundation Rule**
  - *"A RDBMS must manage its stored data using only its relational capabilities"*
- Rule 1: **Information Rule**
  - *"All information in the database should be represented in one and only one way – as values in a table"*
- Rule 2: **Guaranteed Access Rule**
  - *"Each and every datum (atomic value) is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name"*

TUS

Rules of Relational Databases

- Rule 3: **Systematic Treatment of Null Values**
  - *"Null values (which are distinct from empty character strings, strings of blank characters, zeros or any other number) are supported in the fully relational DBMS for representing missing information in a systematic way that is independent of data type"*

TUS

Rules of Relational Databases

- Rule 3: **Systematic Treatment of Null Values**
  - *"Null values (which are distinct from empty character strings, strings of blank characters, zeros or any other number) are supported in the fully relational DBMS for representing missing information in a systematic way that is independent of data type"*
- Rule 4: **Dynamic Online Catalogue Based on the Relational Model**
  - *"The database description is represented at the logical level in the same way as ordinary data, so authorised users can apply the same relational language to its interrogation as they apply to regular data"*

TUS

Rules of Relational Databases

- Rule 5: **Comprehensive Data Sub-language Rule**
  - *"A relational system may support several languages and various modes of terminal use. However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and whose ability to support all of the following is comprehensible:*
    - *Data definition*
    - *View definition*
    - *Data manipulation (interactive and by program)*
    - *Integrity constraints*
    - *Authorisation*
    - *Transaction boundaries (begin, commit and rollback)"*

TUS

Rules of Relational Databases

- Rule 6: **View Updating Rule**
  - “*All views that are theoretically up-dateable are also up-dateable by the system*”

TUS

Rules of Relational Databases

- Rule 6: **View Updating Rule**
  - “*All views that are theoretically up-dateable are also up-dateable by the system*”
- Rule 7: **High-Level Insert, Update and Delete**
  - “*The ability to handle a base relation or a derived relation as a single operand applies not only to the retrieval of data, but also to the insertion, update and deletion of data*”

TUS

Rules of Relational Databases

- Rule 6: **View Updating Rule**
  - “*All views that are theoretically up-dateable are also up-dateable by the system*”
- Rule 7: **High-Level Insert, Update and Delete**
  - “*The ability to handle a base relation or a derived relation as a single operand applies not only to the retrieval of data, but also to the insertion, update and deletion of data*”
- Rule 8: **Physical Data Independence**
  - “*Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods”*

TUS

Rules of Relational Databases

- Rule 9: **Logical Data Independence**
  - “*Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind – and those that theoretically permit unimpairment – are made to the base tables*”

TUS

Rules of Relational Databases

- Rule 9: **Logical Data Independence**
  - "*Application programs and terminal activities remain logically unimpaired when information-preserving changes of any kind – and those that theoretically permit unimpairment – are made to the base tables*"
- Rule 10: **Integrity Independence**
  - "*Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalogue, not in the application programs*"

TUS

Rules of Relational Databases

- Rule 11: **Distribution Independence**
  - "*The data manipulation sub-language of a relational DBMS must enable application programs and terminal activities to remain logically unimpaired whether and whenever data are physically centralised or distributed*"

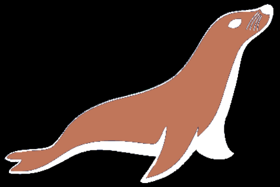
TUS

Rules of Relational Databases

- Rule 11: **Distribution Independence**
  - "*The data manipulation sub-language of a relational DBMS must enable application programs and terminal activities to remain logically unimpaired whether and whenever data are physically centralised or distributed*"
- Rule 12: **Non-subversion**
  - "*If a relational system has or supports a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity rules or constraints expressed in the higher-level (multiple-records-at-a-time) relational language*"

TUS

Fundamental Database Concepts



TUS

Key database concepts

- Tables
- Rows
- Columns
- **Relationships**
  - **Primary Key**
  - **Foreign Key**

Key database concepts – Primary key

Primary Key

orderID	customerID	productID	orderDate	status
1	1	1	2023-10-04	pending
2	2	2	2023-10-05	processing
3	3	3	2023-10-06	delivered

Key database concepts – Primary and Foreign keys

Primary Key

Foreign Keys

orderID	customerID	productID	orderDate	status
1	1	1	2023-10-04	pending
2	2	2	2023-10-05	processing
3	3	3	2023-10-06	delivered

Key database concepts – Primary and Foreign keys

Primary Key

Foreign Keys

orderID	customerID	productID	orderDate	status
1	1	1	2023-10-04	pending
2	2	2	2023-10-05	processing
3	3	3	2023-10-06	delivered

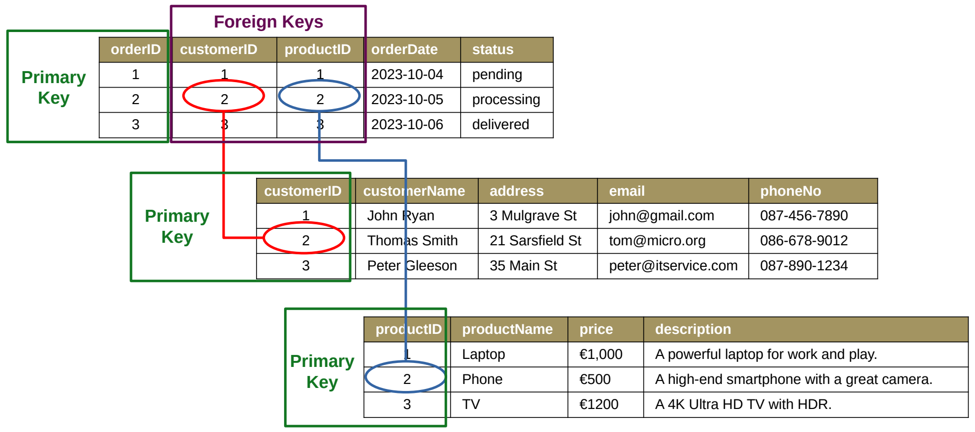
Primary Key

customerID	customerName	address	email	phoneNo
1	John Ryan	3 Mulgrave St	john@gmail.com	087-456-7890
2	Thomas Smith	21 Sarsfield St	tom@micro.org	086-678-9012
3	Peter Gleeson	35 Main St	peter@itservice.com	087-890-1234

Primary Key

productID	productName	price	description
1	Laptop	€1,000	A powerful laptop for work and play.
2	Phone	€500	A high-end smartphone with a great camera.
3	TV	€1200	A 4K Ultra HD TV with HDR.

Key database concepts – Relationships



orderID	customerID	productID	orderDate	status
1	1	1	2023-10-04	pending
2	2	2	2023-10-05	processing
3	3	3	2023-10-06	delivered

customerID	customerName	address	email	phoneNo
1	John Ryan	3 Mulgrave St	john@gmail.com	087-456-7890
2	Thomas Smith	21 Sarsfield St	tom@micro.org	086-678-9012
3	Peter Gleeson	35 Main St	peter@itservice.com	087-890-1234

productID	productName	price	description
1	Laptop	€1,000	A powerful laptop for work and play.
2	Phone	€500	A high-end smartphone with a great camera.
3	TV	€1200	A 4K Ultra HD TV with HDR.

Key database concepts

- Tables
- Rows
- Columns
- Relationships
  - Primary Key
  - Foreign Key
- **Structured Query Language (SQL)**

Structured Query Language (SQL)

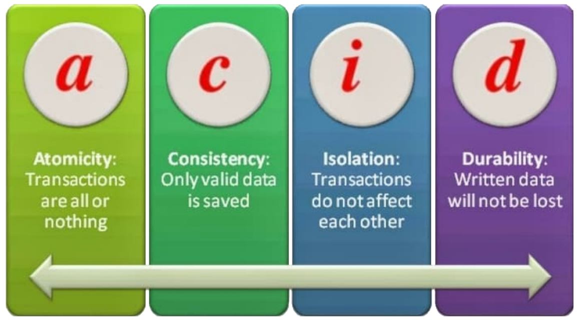
```
SQL> SELECT
  o.orderID,
  c.customerName,
  p.productName,
  o.orderDate,
  o.status
FROM orders o
JOIN customers c ON c.customerID = o.customerID
JOIN products p ON p.productID = o.productID;

+---------+---------------+-------------+------------+------------+
| orderID | customerName  | productName | orderDate  |   status   |
+---------+---------------+-------------+------------+------------+
|    1    | John Ryan     | Laptop      | 2023-10-04 | pending    |
|    2    | Thomas Smith  | Phone       | 2023-10-05 | processing |
|    3    | Peter Gleeson | TV          | 2023-10-06 | delivered  |
+---------+---------------+-------------+------------+------------+
```

Key database concepts

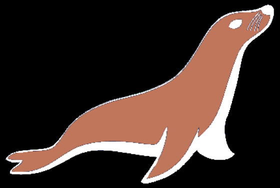
- Indexes
- Schema
- Normalisation
  - There is no data redundancy
  - Data dependencies are logical
- Constraints
- Transactions
  - Commits
  - Rollbacks
- Locking

Atomicity, Consistency, Isolation, Durability

- Essential properties of any RDBMS are:


a
Atomicity: Transactions are all or nothing
c
Consistency: Only valid data is saved
i
Isolation: Transactions do not affect each other
d
Durability: Written data will not be lost

The Database Market



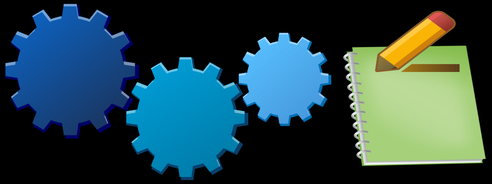
TUS

RDBMS Market

ORACLE
MySQL
MariaDB
PostgreSQL
IBM
DB2
Microsoft
SQL Server

Laboratory #1

Building Data Models



TUS

Learning Objectives

- Distinguish Data Modelling Tools ✓
- Write simple File-Based Data Models ✓
- Classify different database types ✓
- List the 12 Rules of Relational Databases ✓
- Discuss Fundamental Database Concepts - ACID ✓
- List databases available on the market ✓

TUS

TUS
Ollscoil Teicneolaíochta na Sionainne: Lár Tíre, An tIarthar Láir
Technological University of the Shannon: Midlands Midwest

EUR ING Dr Diarmuid Ó Briain
Innealtóir Cairte agus
Léachtóir Sinsearach

E diarmuid.obriain@tus.ie | W tus.ie
Campas Maoilis, Páirc Maoilis,
Luimneach, V94 EC5T, Éire

CEng, FIEI
CISSP

Thank you

TUS