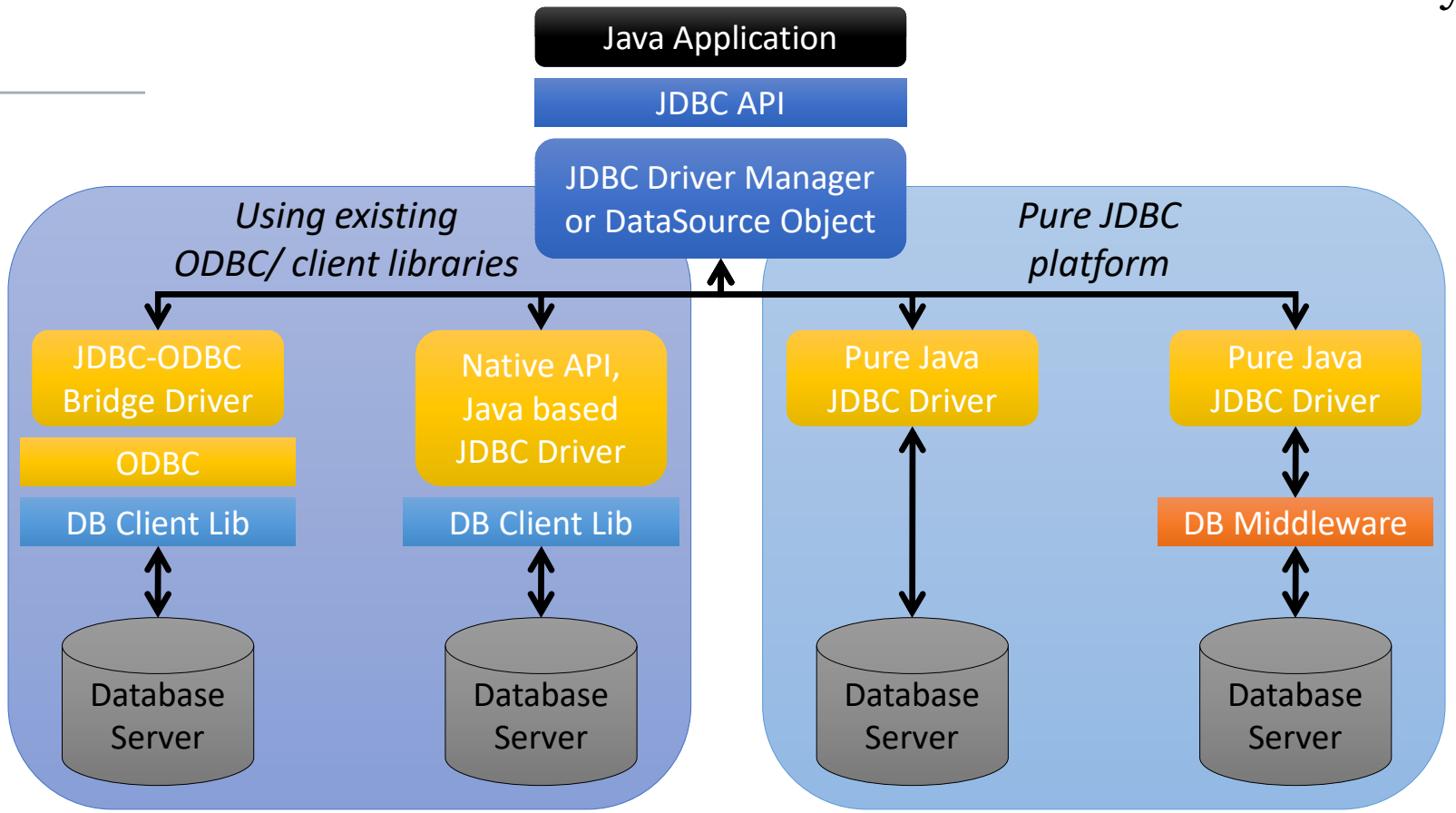
Lancaster University

# SCC201: DATABASES

Week 6: JDBC and Revision

http://java.sun.com/products/jdbc/datasheet.html

# The Four JDBC Drivers



# Driver Manager Tasks

Lancaster University

- Register Database
- Establish Connection
- Search JDBC Driver
- Connection Pooling

# JDBC excep

- Now
  and i
- See p
- JDBC
  *java.*
  *javax*

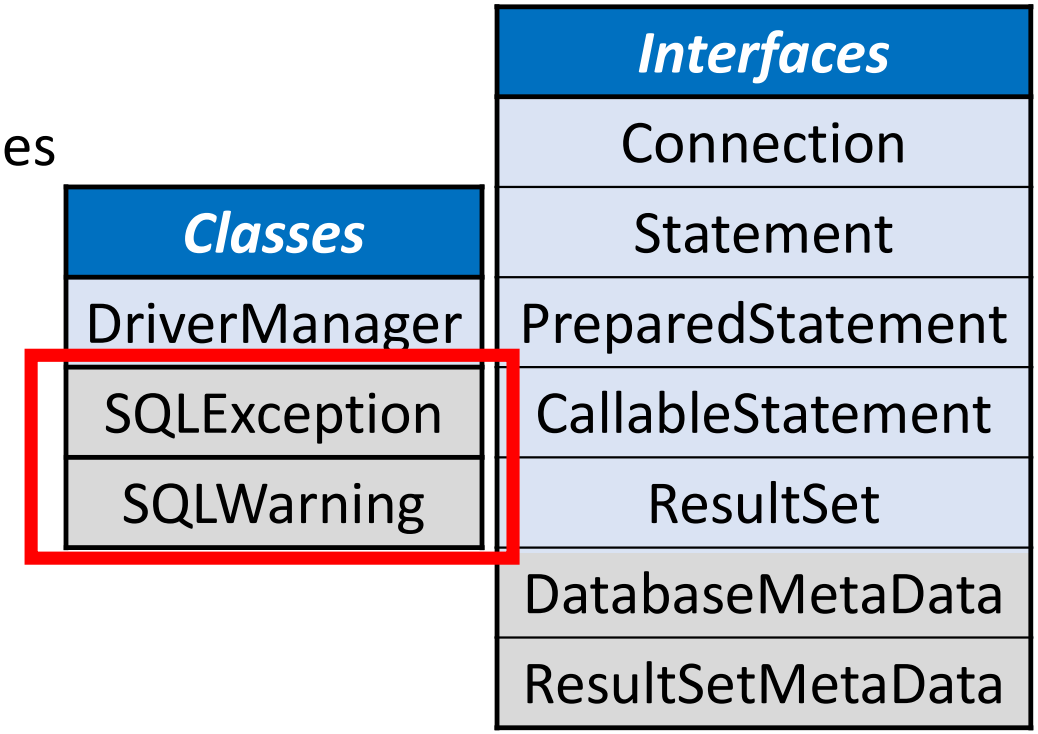
```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        String username = "your_username";
        String password = "your_password";

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

| ces |
| --- |
| tion |
| ent |
| atement |
| tement |
| Set |
| etaData |
| etaData |

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*



JDBC
excep

- Now
and i
- See p
- JDBC
*java..*
*javax*

ster
sity

| ces |
| --- |
| tion |
| ent |
| atement |
| tement |
| Set |
| etaData |
| etaData |

DATABASE.java
~/Desktop

Open

Save

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

# JDBC excep

- Now and i
- See p
- JDBC *java.. javax*

DATABASE.java
~/Desktop

Open | Save

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

| ces |
| --- |
| tion |
| ent |
| atement |
| tement |
| Set |
| etaData |
| etaData |

JDBC
excep

- Now
and i
- See p
- JDBC
*java..*
*javax*

DATABASE.java
~/Desktop

Open Save

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

ster
sity

| ces |
| --- |
| tion |
| ent |
| atement |
| tement |
| Set |
| etaData |
| etaData |

# JDBC excep

- Now and i
- See p
- JDBC *java.. javax*

DATABASE.java
~/Desktop

Open Save

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
turker@vdi-scc201-006:~/Desktop$ java -cp ".:/usr/share/java/mariadb-java-client.jar:" DATABASE.java
java.sql.SQLTransientConnectionException: (conn=21) Can't create database 'SCC201COURSEWORK'; database exists
```

| ces |
| --- |
| tion |
| ent |
| atement |
| tement |
| Set |
| etaData |
| etaData |

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

| *Classes* |
| --- |
| DriverManager |
| SQLException |
| SQLWarning |

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {
    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        String username = "your_username";
        String password = "your_password";

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Check if the database already exists
            ResultSet resultSet = connection.getMetaData().getCatalogs();
            boolean databaseExists = false;
            while (resultSet.next()) {
                if (resultSet.getString(1).equalsIgnoreCase(databaseName)) {
                    databaseExists = true;
                    break;
                }
            }
            resultSet.close();
```

***<protocol>:<subprotocol>:<subname>***

Lancaster University

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

# In your case, you will not add a password and username.

Lancaster University

Open DATABASE.Java ~/Desktop Save

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement()
        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

| *Classes* |
| --- |
| DriverManager |
| SQLException |
| SQLWarning |

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

# JDBC API: interfaces, classes and exceptions

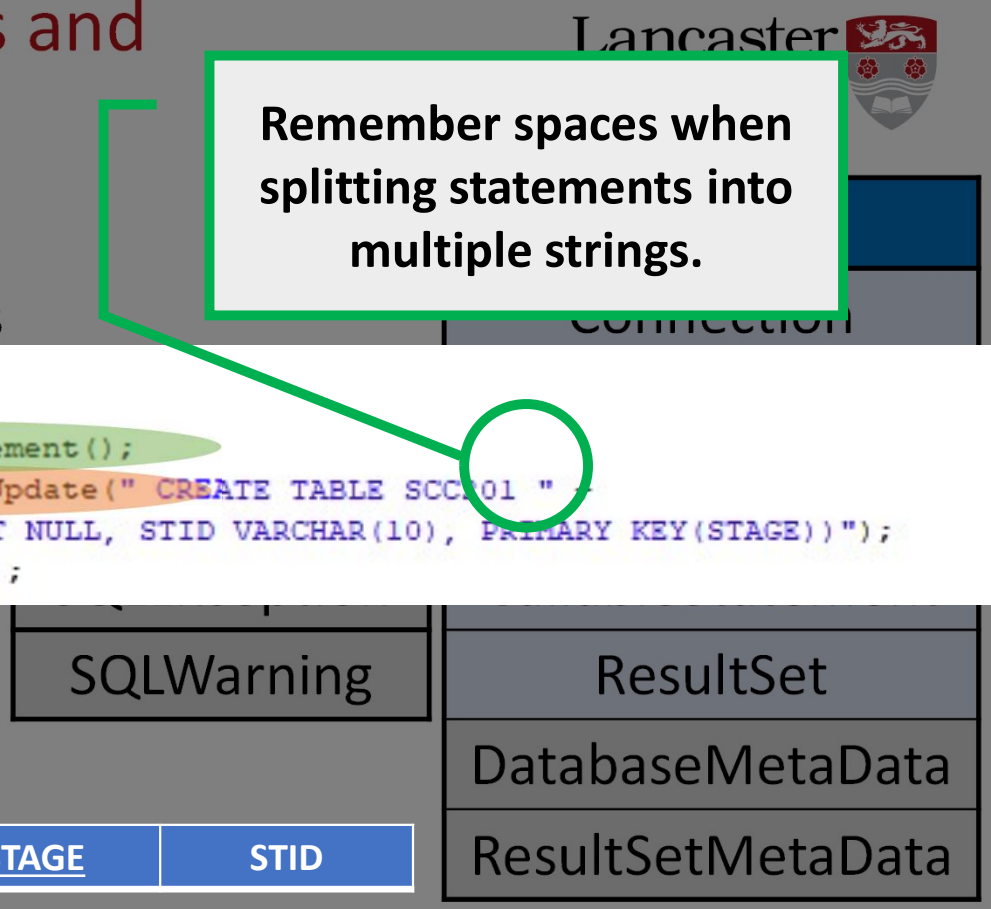
Lancaster

**Remember spaces when splitting statements into multiple strings.**

- Now look at some of these classes

```
try {

    Statement UrazsOrder = conn.createStatement();
    int resultOfQuery = UrazsOrder.executeUpdate(" CREATE TABLE SCC201 " +
    "(STNAME VARCHAR(32), STAGE INTEGER NOT NULL, STID VARCHAR(10), PRIMARY KEY(STAGE))");
     System.out.println("Table is created");
```

- JDBC API available in packages *java.sql* and *javax.sql*




| STNAME | STAGE | STID |
|---|---|---|

# Inserting into a table

Lancaster University

What would happen if, instead of 10, we had 1.000.000?

Compilation of this statement takes 2 millisecond,

```
public void insertStatement(String STNAME, int  stage, String STID )
{
        Connection conn = null;
        Scanner getDBNameObj = new Scanner(System.in);
        System.out.println("ManualInsert");
       try {
               conn = this.connect();
               System.out.println("Enter the name of the Table to STATEMENT INSERT");
               String TBName = getDBNameObj.nextLine();
               for(int i = 0 ; i < 10; i++)
               {
                       Statement UrazInserts = conn.createStatement();
                       UrazInserts.executeUpdate("INSERT INTO "+TBName+" (STNAME, STAGE, STID) VALUES ( \' " + STNAME +" \', " + stage+i +" ,\'" + STID+"\' );");

               }
       }catch (SQLException e) {
           System.out.println(e.getMessage());
       } finally { try {

           if (conn != null) {
               conn.close();
           }
       } catch (SQLException ex) {
           System.out.println(ex.getMessage());
               }}

}
```

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail.
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

| *Classes* |
| --- |
| DriverManager |
| SQLException |
| SQLWarning |

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

# Prepared Statements

Lancaster University

- Allows dynamic and parametric queries.
- During the initialisation, it introduces some overheads compared to Statements. So, use PreparedStatements if you have many iterations.
- Prevents SQL Injection Attacks in JAVA.

**Assume query:** *Retrieve the id, firstname and lastname from authors where* firstname is evil'ex and lastname is Newman
**Then the query string becomes:**
*SELECT id, firstname, lastname FROM authors WHERE firstname = 'evil'ex' and lastname ='newman';*
**which the database will return:** Incorrect syntax near il' as the database tried to execute evil.

# JDBC AP exceptio

- Now look and inter
- See pp 1(
- JDBC API *java.sql* *javax.sql*

```
public void insert(String STNAME, int  stage, String STID )
{
   Scanner getDBNameObj = new Scanner(System.in);
   Connection conn = null;
   System.out.println("InsertTable");
   try
   {
       conn = this.connect();
       System.out.println("Connection to MySQL Database has been established to insert to Table.");

       System.out.println("Enter the name of the Table to PREPARED STATEMENT INSERT");
       String TBName = getDBNameObj.nextLine();
       String sql = "INSERT INTO "+TBName+"(STNAME, STAGE, STID) VALUES(?,?,?)";
       PreparedStatement pstmt = conn.prepareStatement(sql);
       for(int i = 0 ; i < 1000 ; i++)
       {
           pstmt.setString(1, STNAME);
           pstmt.setInt(2, stage+i);
           pstmt.setString(3, STID);
           pstmt.executeUpdate();
       }
   }
   catch (SQLException e)
   {
       System.out.println(e.getMessage());
   }
   finally
   {
       try {
           if (conn != null)
           {
               conn.close();
               System.out.println("Connection to MySQL is cut.");
           }
       }
       catch (SQLException ex)
       {
           System.out.println(ex.getMessage());
       }
   }
}
```

1 2 3

| STNAME | STAGE | STID |
|---|---|---|

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail.
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

| *Classes* |
| --- |
| DriverManager |
| SQLException |
| SQLWarning |

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

# JDBC API: interfaces, classes and excepti

Lancaster

- Now lo
  and inte
- See pp
- JDBC A
  *java.sq*
  *javax.s*

| STNAME | STAGE | STID |
|---|---|---|

```
try {
    System.out.println("PrintTabe");
   conn = this.connect();

    Statement UrazsQuery = conn.createStatement();
    ResultSet UrazsAnswer = UrazsQuery.executeQuery("SELECT * FROM SCC201; ");
    if(UrazsAnswer.next()==false)
        System.out.println("Empty Table");
    else
    {
        String stname = UrazsAnswer.getString(1);
        int stage =  UrazsAnswer.getInt("STAGE");
        String stid = UrazsAnswer.getString(3);
        System.out.println(stname+" " + stage + " " + stid);
        while(UrazsAnswer.next())
        {
            stname = UrazsAnswer.getString(1);
            stage =  UrazsAnswer.getInt(2);
            stid = UrazsAnswer.getString(3);
            System.out.println(stname+" " + stage + " " + stid);
        }
    }


}
```

| ces |
|---|
| ction |
| nent |
| atement |
| atement |
| Set |
| letaData |
| letaData |

# JDBC API: interfaces, classes and exceptions

Lancaster University

- Now look at some of these classes and interfaces in more detail.
- See pp 1004-1010, C&B 5th Ed.
- JDBC API available in packages *java.sql* and *javax.sql*

| *Classes* |
| --- |
| DriverManager |
| SQLException |
| SQLWarning |

| *Interfaces* |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

```java
import java.sql.*;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();


        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            DatabaseMetaData dbmd = connection.getMetaData();
            System.out.println("Database '" + databaseName + "' created successfully.");

            System.out.println("Driver Name: "+dbmd.getDriverName());
            System.out.println("Driver Version: "+dbmd.getDriverVersion());
            System.out.println("User Name: "+dbmd.getUserName());
            System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
            System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Lancaster University

| Interfaces |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

```java
import java.sql.*;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try (
            // Establishing a connection to the MySQL server
            Connection connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();


        ) {
            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            DatabaseMetaData dbmd = connection.getMetaData();
            System.out.println("Database '" + databaseName + "' created successfully.");

            System.out.println("Driver Name: "+dbmd.getDriverName());
            System.out.println("Driver Version: "+dbmd.getDriverVersion());
            System.out.println("User Name: "+dbmd.getUserName());
            System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
            System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Lancaster University

| Interfaces |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

```java
import java.sql.*;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";


        // Database name to be created
        String databaseName = "SCC201COURSEWORK";
```

```
turker@vdi-scc201-039:~/Desktop$ java -cp ".:/usr/share/java/mariadb-java-client
.jar:" DATABASE.java
Database 'SCC201COURSEWORK' created successfully.
Driver Name: MariaDB Connector/J
Driver Version: 2.7.4
User Name: null
Database Product Name: MySQL
Database Product Version: 8.0.35-0ubuntu0.22.04.1
turker@vdi-scc201-039:~/Desktop$
```

```java
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            DatabaseMetaData dbmd = connection.getMetaData();
            System.out.println("Database '" + databaseName + "' created successfully.");

            System.out.println("Driver Name: "+dbmd.getDriverName());
            System.out.println("Driver Version: "+dbmd.getDriverVersion());
            System.out.println("User Name: "+dbmd.getUserName());
            System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
            System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Lancaster University

| Interfaces |
| --- |
| Connection |
| Statement |
| PreparedStatement |
| CallableStatement |
| ResultSet |
| DatabaseMetaData |
| ResultSetMetaData |

# Keeping Things Tidy

Lancaster University

- Database objects can tie up lots of resources
  - Always close and as soon as possible use finally { } to ensure resources are always freed even in the event of an exception.

```
try {
    /*
     *    Do some work
     */
} finally {
    /*
     *    Free resources
     */
}
```

```
try {
    //    Do some work
} catch ( Exception e ) {
    //    Handle exception
} finally {
    //    Free resources
}
```

# JDBC AP exceptio

- Now loo and inte
- See pp 1
- JDBC AP *java.sql* *javax.sq*

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        Connection connection = null;

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try {
            // Establishing a connection to the MySQL server
            connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();

            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally
        {
            try{
                if(connection !=null){
                  connection.close();
                }
                } catch (SQLException ex){
                        System.out.println(ex.getMessage());
                }
        }
    }
}
```

lancaster university

erfaces

nection

tement

dStatement

eStatement

sultSet

eMetaData

etMetaData

# JDBC AP
# exceptio

- Now loo
  and inte
- See pp 1
- JDBC AP
  *java.sql*
  *javax.sq*

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        Connection connection = null;

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try {
            // Establishing a connection to the MySQL server
            connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();

            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally
        {
            try{
                if(connection !=null){
                  connection.close();
                }
                } catch (SQLException ex){
                        System.out.println(ex.getMessage());
                }
        }
    }
}
```

lancaster university

**erfaces**

- nection
- tement
- dStatement
- eStatement
- sultSet
- eMetaData
- etMetaData

# JDBC AP
# exceptio

- Now loo
  and inte
- See pp 1
- JDBC AP
  *java.sql*
  *javax.sq*

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        Connection connection = null;

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try {
            // Establishing a connection to the MySQL server
            connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();

            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally
        {
            try{
                if(connection !=null){
                  connection.close();
                }
                } catch (SQLException ex){
                        System.out.println(ex.getMessage());
                }
        }
    }
}
```



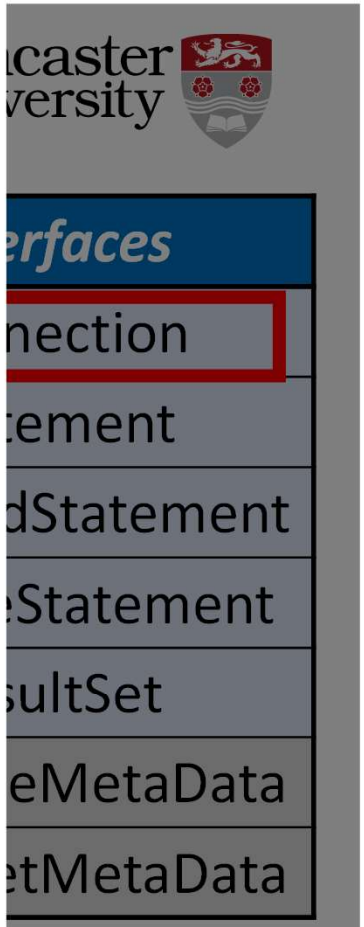


erfaces
nection
tement
dStatement
eStatement
sultSet
eMetaData
etMetaData

# JDBC AP
# exceptio

- Now loo
  and inte
- See pp 1
- JDBC AP
  *java.sql*
  *javax.sq*

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateDatabase {

    public static void main(String[] args) {
        // JDBC connection parameters
        String jdbcUrl = "jdbc:mysql://localhost:3306/";
        Connection connection = null;

        // Database name to be created
        String databaseName = "SCC201COURSEWORK";

        try {
            // Establishing a connection to the MySQL server
            connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object for executing SQL commands
            Statement statement = connection.createStatement();

            // Creating the database
            String createDatabaseQuery = "CREATE DATABASE " + databaseName;
            statement.executeUpdate(createDatabaseQuery);

            System.out.println("Database '" + databaseName + "' created successfully.");

        } catch (SQLException e) {
            e.printStackTrace();
        } finally
        {
            try{
                if(connection !=null){
                  connection.close();
                }
                } catch (SQLException ex){
                        System.out.println(ex.getMessage());
                }
        }
    }
}
```



# Summary

Lancaster University

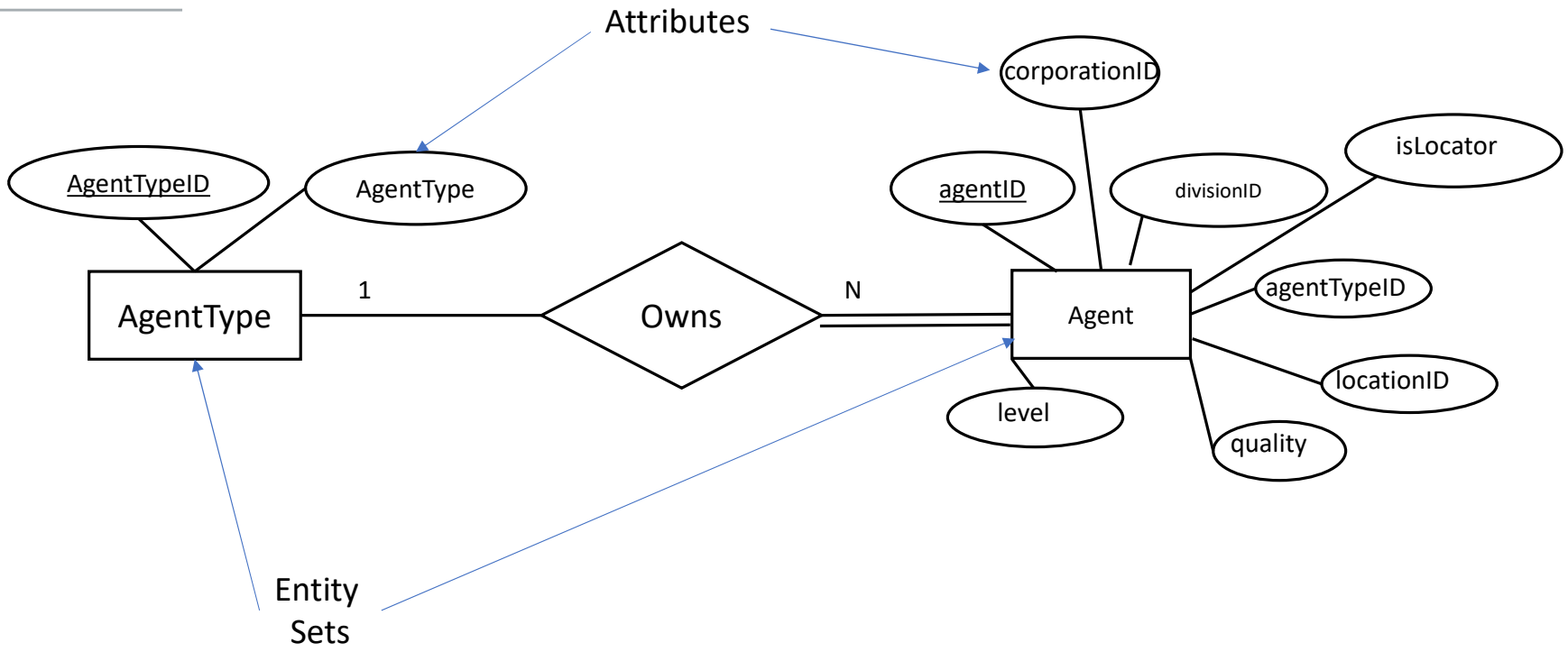
- JDBC's goal is:
  - DBMS-independent interface.
  - Can access any data source without recoding
- JDBC supports basic SQL functionality
- JDBC is flexible, easy to manage and inexpensive

# Revision (Exercises will appear!)

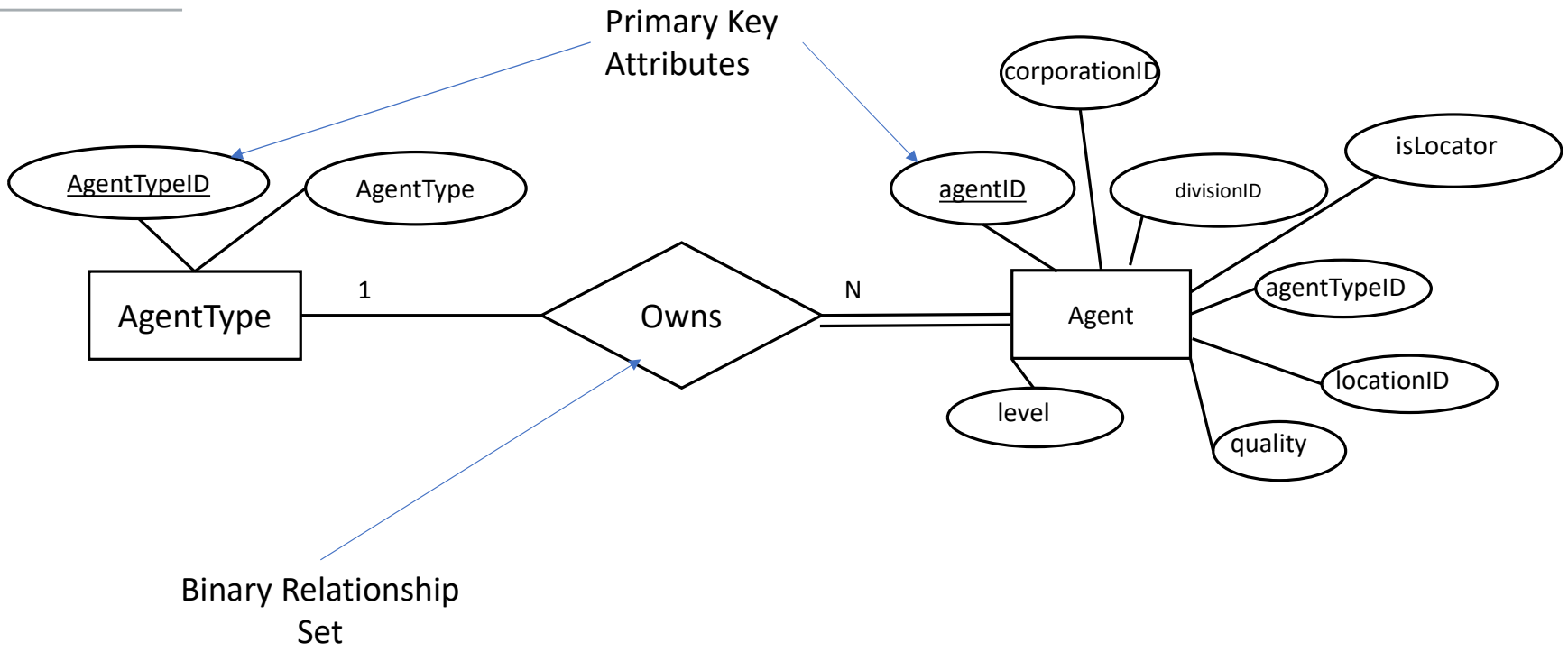
Lancaster University

- Week 1
  - Database, Table, DBMS
  - Primary Key, Candidate Key, different types of Attributes
  - Foreign Keys
  - Participation, Multiplicity constraints
  - Referential Integrity
  - Notations for the ERD.

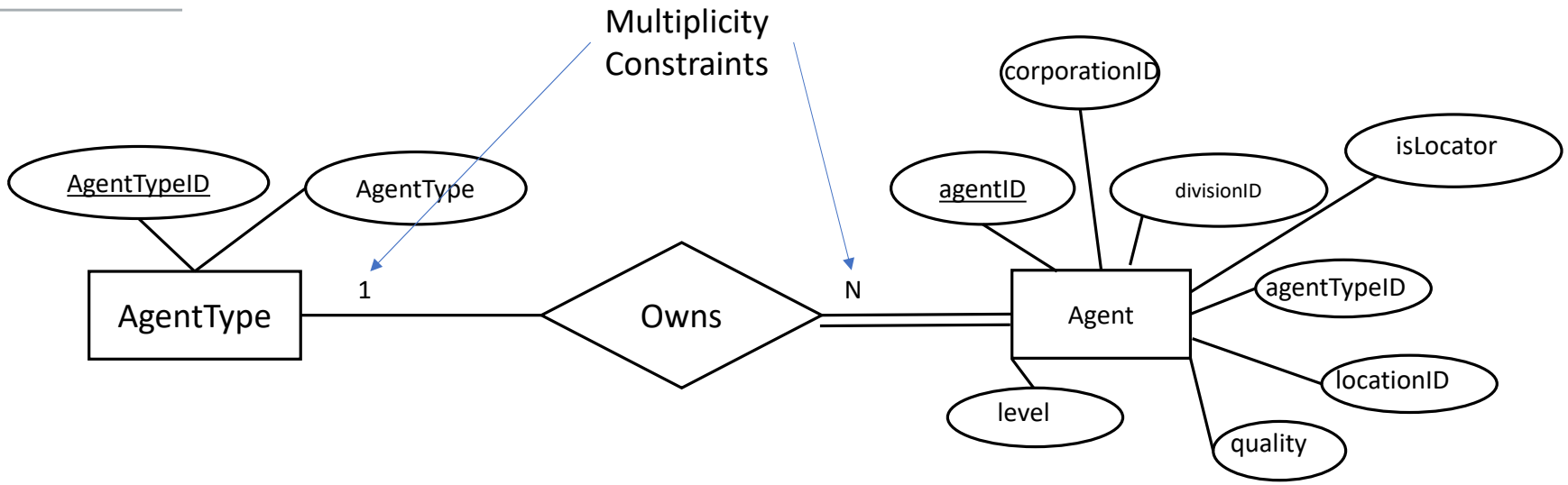
# ER Diagram Symbols



# ER Diagram Symbols



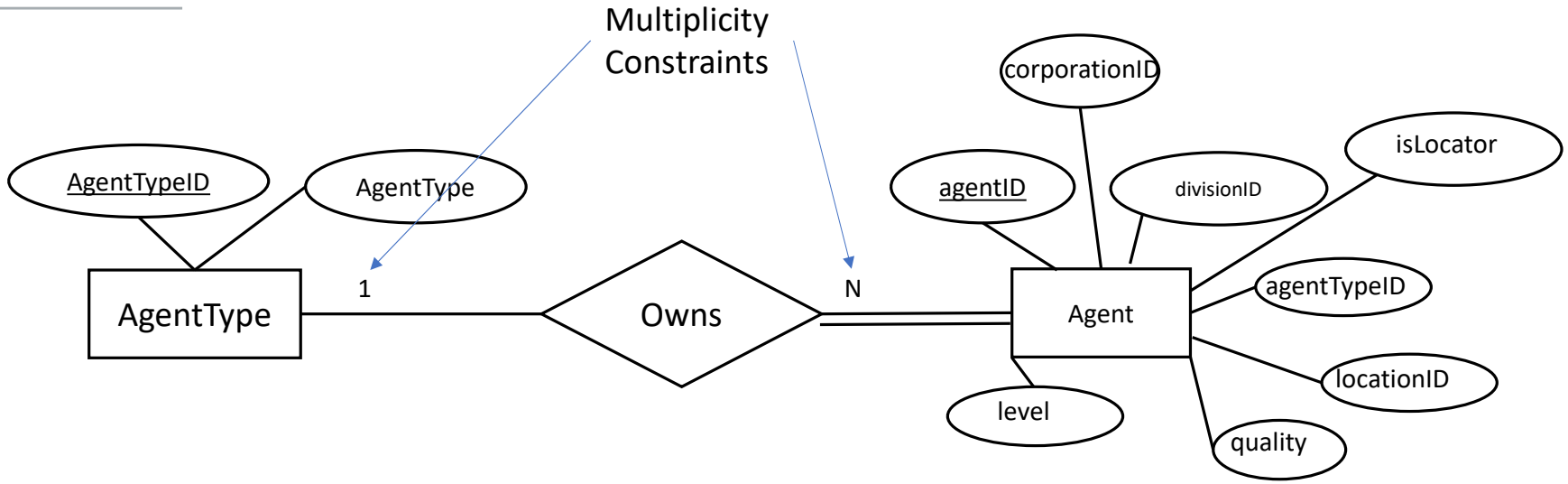
# ER Diagram Symbols



# ER Diagram Symbols

In any given binary relationship, we consider the pairings of primary key values of the entity sets.

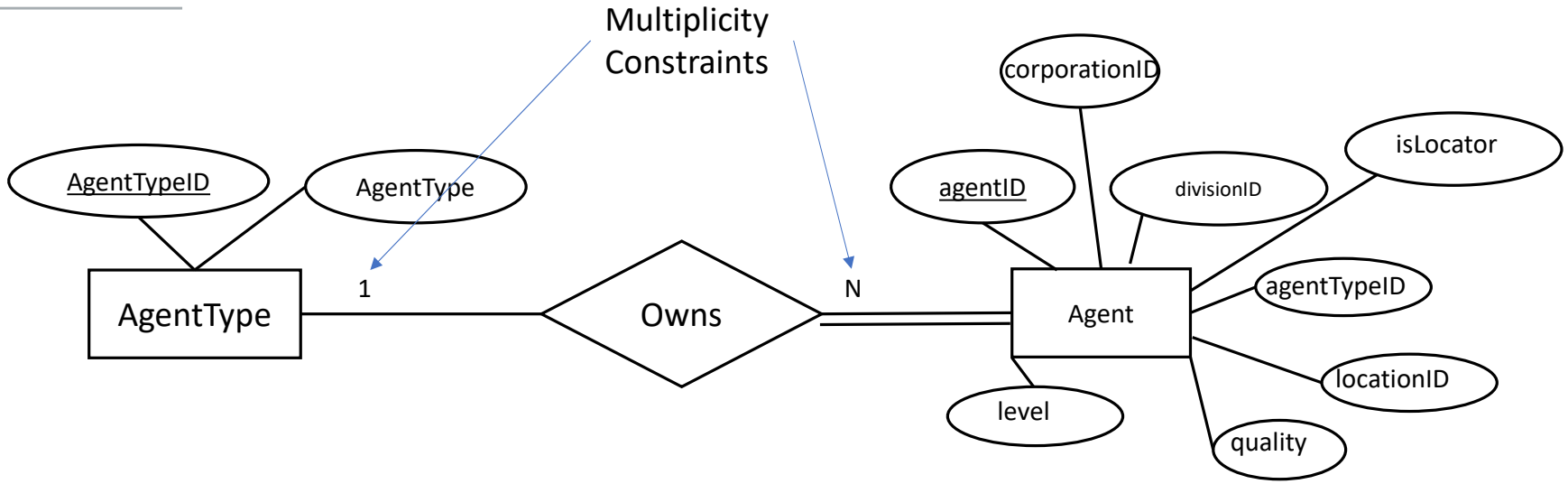
AgentTypeID – agentID



# ER Diagram Symbols

In any given binary relationship, we consider the pairings of primary key values of the entity sets.

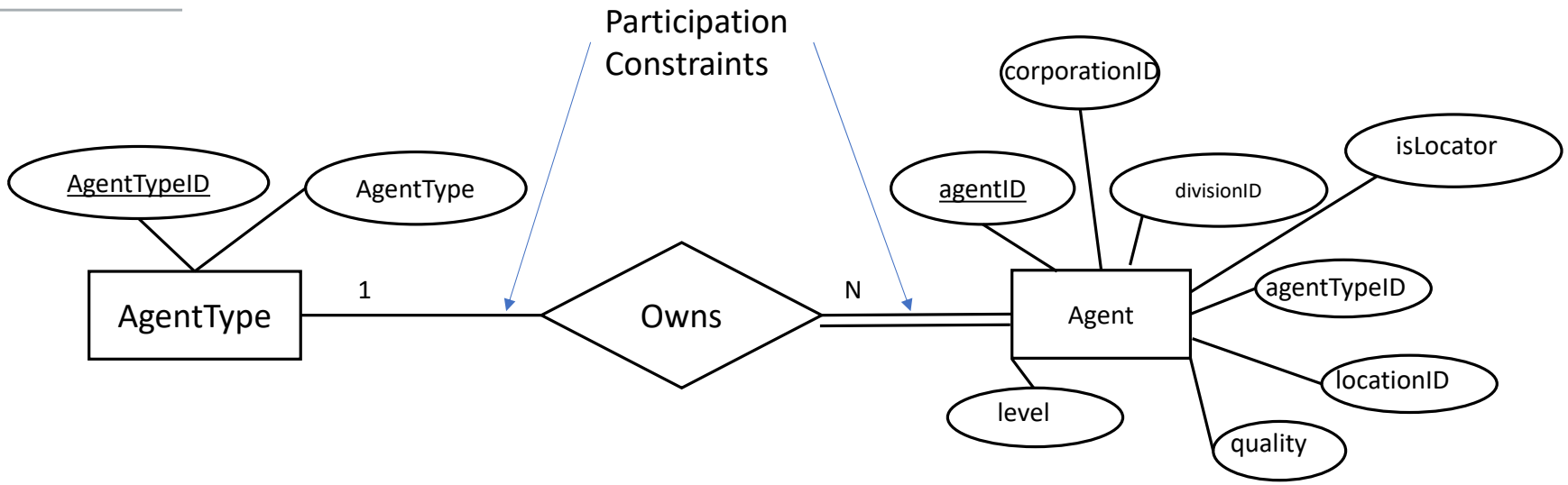
agentTypeID – agentID




If AgentTypeID can relate to multiple agentID's, its multiplicity constraint is MANY (N is written on the opposite end of the relationship set). Otherwise, it is one (1).

If agentID can relate to more than one AgentTypeID, its multiplicity constraint is MANY (N). Otherwise, it is one (1).

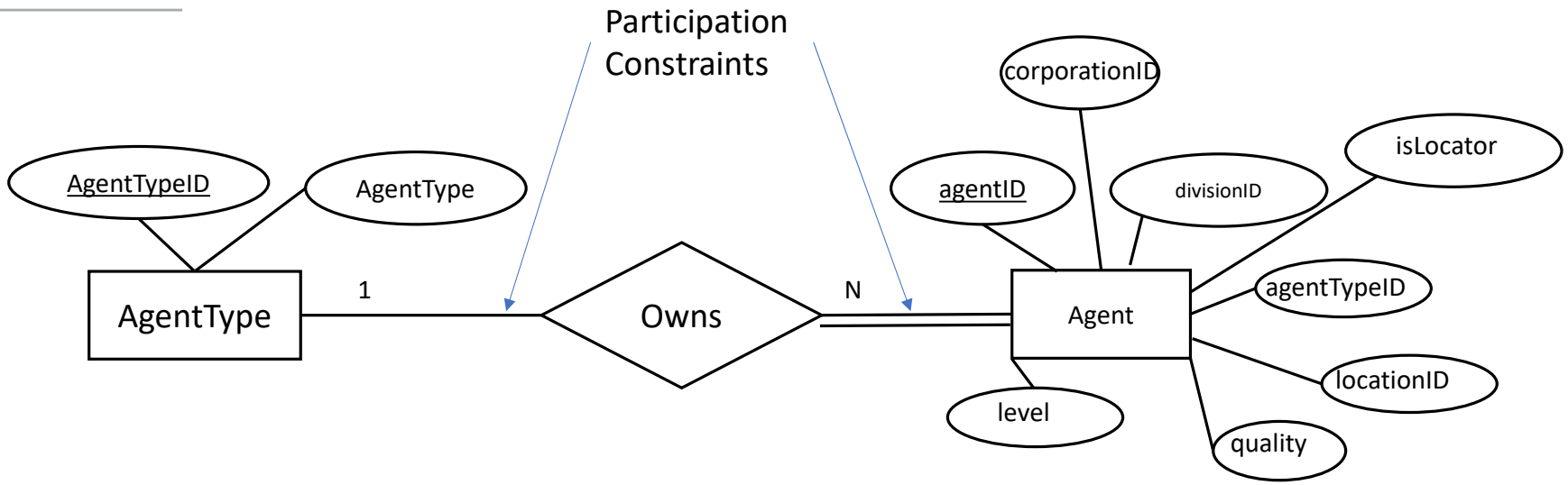
# ER Diagram Symbols



# ER Diagram Symbols

In any given binary relationship, we consider the pairings of primary key values of the entity sets.

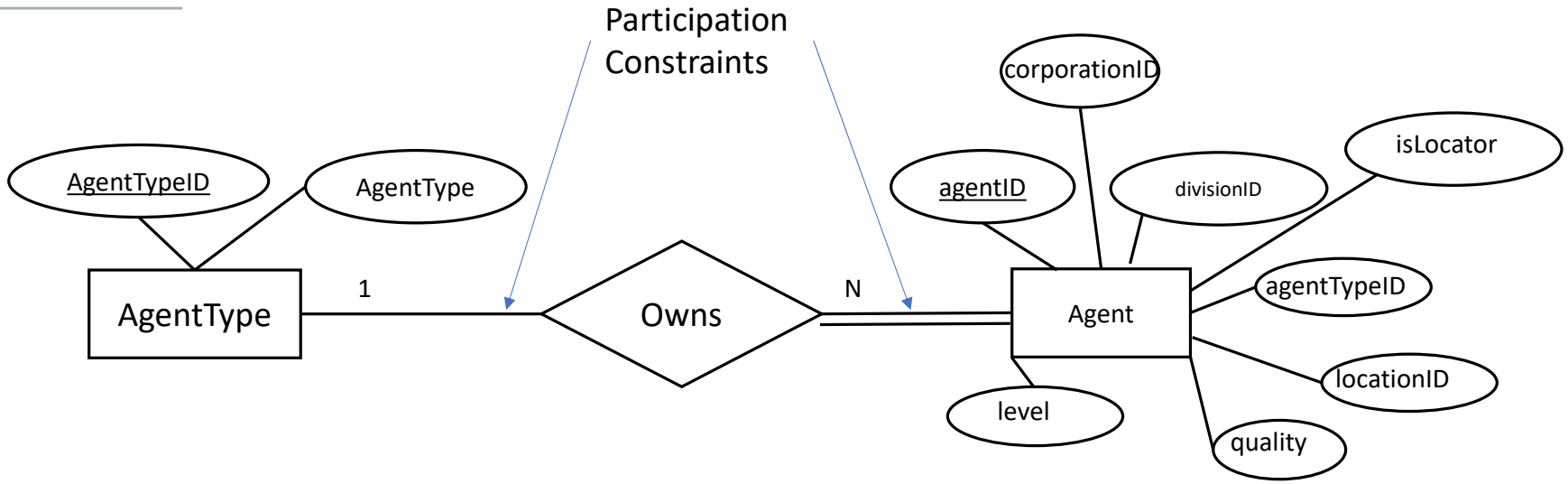
agentTypeID – agentID



# ER Diagram Symbols

In any given binary relationship, we consider the pairings of primary key values of the entity sets.

agentTypeID – agentID




If AgentTypeID must relate to agentIDs, its participation constraint is TOTAL (by the entity's side, double-lined). Otherwise, it is Partial (single-lined).

If agentID must relate to AgentTypeIDs, its participation constraint is TOTAL (by the entity's side double-lined). Otherwise, it is Partial (single-lined).

# Revision

Lancaster University

- Week 2
  - From ERD to Relational Model
  - It is a phase between DDL and ERD
  - Relation has two components
    - Relational Schema
    - Instance
  - Domains and names of relations are set.
  - All the Integrity Constraints : Entity Integrity, Domain, Multiplicity, Participation and Referential Integrity Constraints are set using keywords like:
    - Foreign Key, Primary Key, UNIQUE, ON DELETE/UPDATE (cascade, reject,set null, set default)

# Relational Model

Lancaster University

- Relation:
  - Consist of
    - Relational Schema (the signature of the relation)
    - Instance; a table with data.

# Relation Palbasics from Palworld

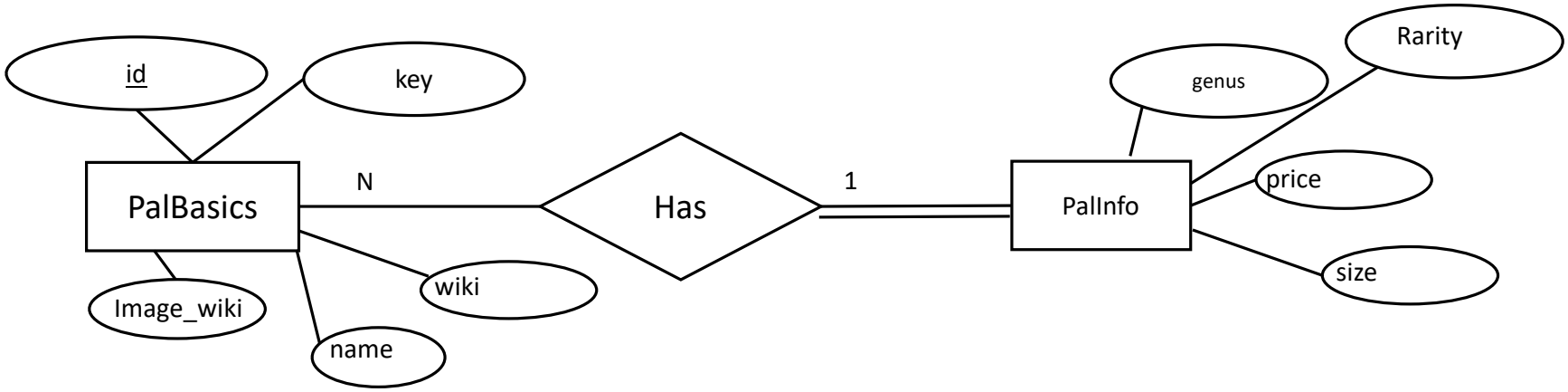

https://github.com/jhideki/palworld-api/tree/master/data

RS: *Palbasics*(id INTEGER, key INTEGER, name TEXT, wiki TEXT, image_wiki TEXT)
PK:key

Domains

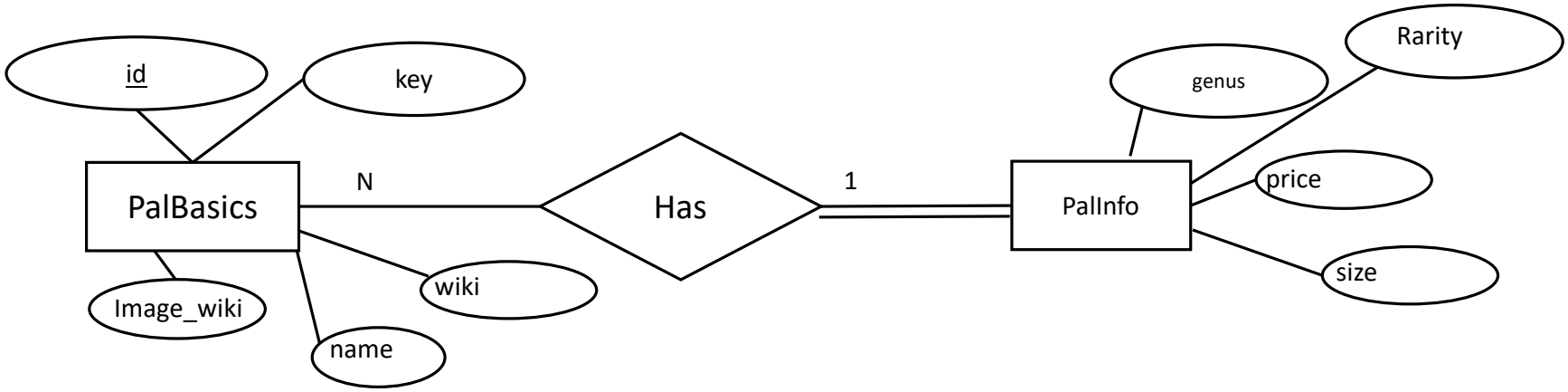
| id | key | name | wiki | image_wiki |
|---|---|---|---|---|
| 1 | 1 | Lamball | https://palworld.fandom.com/wiki/Lamball | https://static.wikia.nocookie.net/palworld/images/0/01/Lamball_menu.png/ |
| 2 | 2 | Cattiva | https://palworld.fandom.com/wiki/Cattiva | https://static.wikia.nocookie.net/palworld/images/5/51/Cattiva_menu.png/ |
| 3 | 3 | Chikipi | https://palworld.fandom.com/wiki/Chikipi | https://static.wikia.nocookie.net/palworld/images/f/f4/Chikipi_menu.png/ |
| 4 | 4 | Lifmunk | https://palworld.fandom.com/wiki/Lifmunk | https://static.wikia.nocookie.net/palworld/images/d/dc/Lifmunk_menu.png/ |
| 5 | 5 | Foxparks | https://palworld.fandom.com/wiki/Foxparks | https://static.wikia.nocookie.net/palworld/images/d/d7/Foxparks_menu.png/ |
| 6 | 6 | Fuack | https://palworld.fandom.com/wiki/Fuack | https://static.wikia.nocookie.net/palworld/images/5/5e/Fuack_menu.png/ |
| 7 | 7 | Sparkit | https://palworld.fandom.com/wiki/Sparkit | https://static.wikia.nocookie.net/palworld/images/c/ce/Sparkit_menu.png/ |
| 8 | 8 | Tanzee | https://palworld.fandom.com/wiki/Tanzee | https://static.wikia.nocookie.net/palworld/images/4/40/Tanzee_menu.png/ |
| 9 | 9 | Rooby | https://palworld.fandom.com/wiki/Rooby | https://static.wikia.nocookie.net/palworld/images/2/21/Rooby_menu.png/ |
| 10 | 10 | Pengullet | https://palworld.fandom.com/wiki/Pengullet | https://static.wikia.nocookie.net/palworld/images/3/38/Pengullet_menu.png/ |
| 11 | 11 | Penking | https://palworld.fandom.com/wiki/Penking | https://static.wikia.nocookie.net/palworld/images/b/b5/Penking_menu.png/ |
| 12 | 12 | Jolthog | https://palworld.fandom.com/wiki/Jolthog | https://static.wikia.nocookie.net/palworld/images/5/52/Jolthog_menu.png/ |
| 13 | 13 | Gumoss | https://palworld.fandom.com/wiki/Gumoss | https://static.wikia.nocookie.net/palworld/images/2/2e/Gumoss_menu.png/ |
| 14 | 14 | Vixy | https://palworld.fandom.com/wiki/Vixy | https://static.wikia.nocookie.net/palworld/images/9/9e/Vixy_menu.png/ |
| 15 | 15 | Hoocrates | https://palworld.fandom.com/wiki/Hoocrates | https://static.wikia.nocookie.net/palworld/images/e/ef/Hoocrates_menu.png/ |

# Multiplicity and Participation Constraints

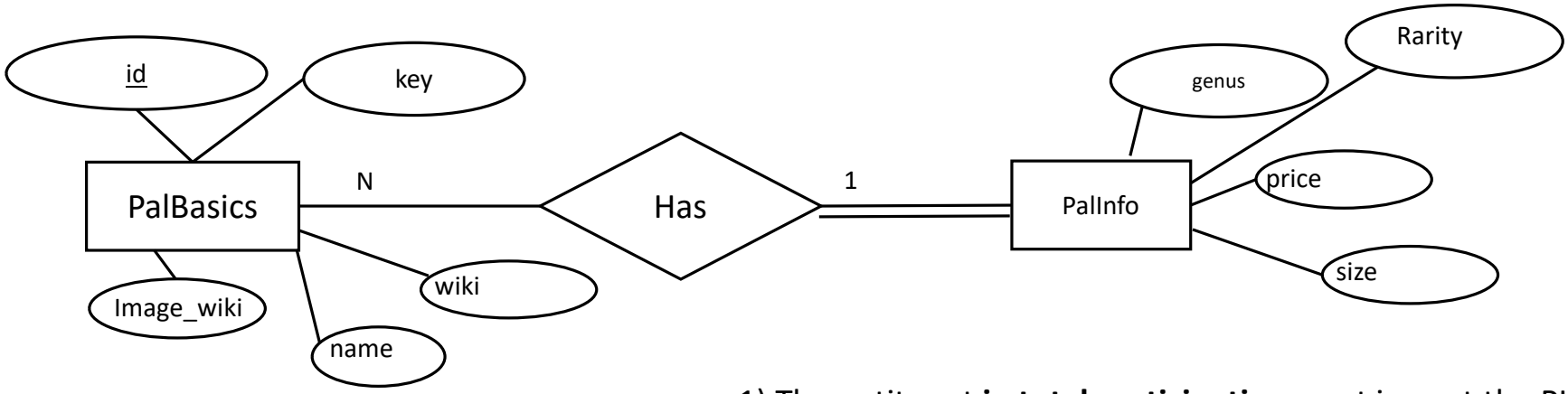
# Multiplicity and Participation Constraints


For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.



# Multiplicity and Participation Constraints

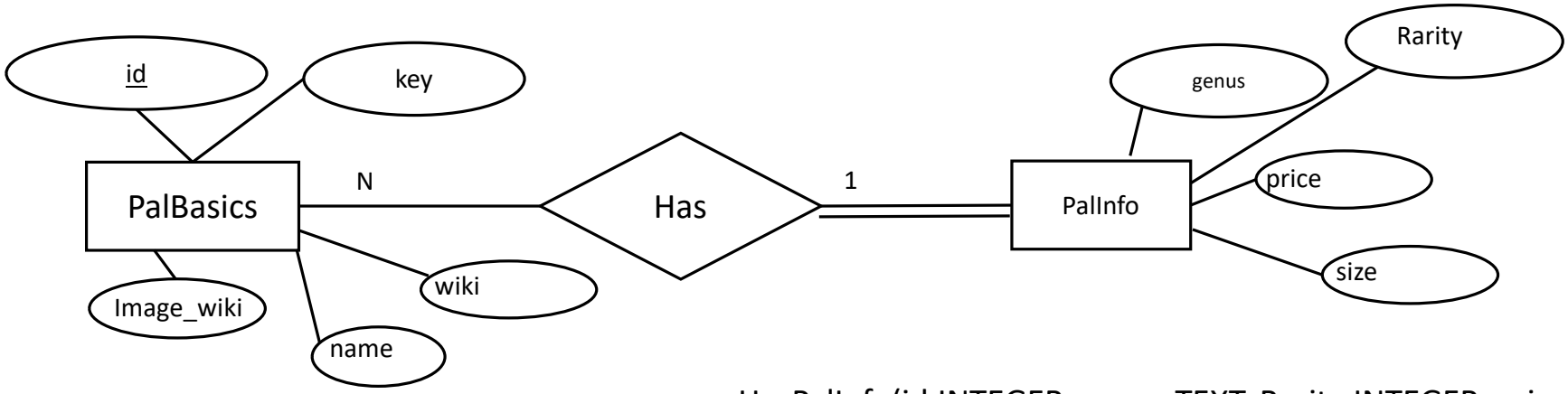

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




1) The entity set **in total participation** must import the PK of the entity set in relation as a FOREIGN KEY

# Multiplicity and Participation Constraints

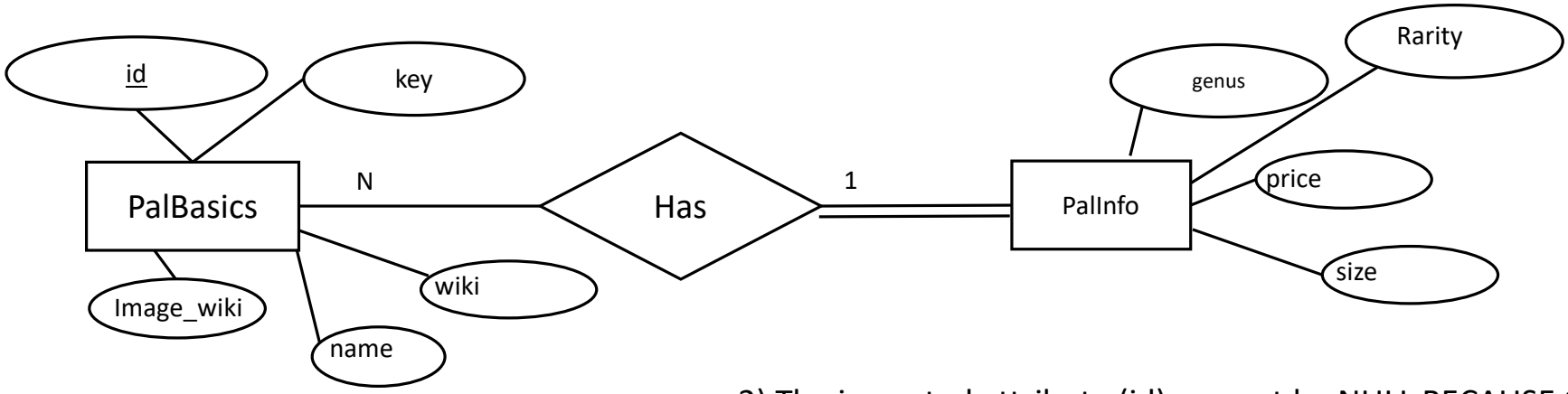

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




HasPalInfo(id INTEGER, genus TEXT, Rarity INTEGER, price FLOAT, size INTEGER)
FOREIGN KEY id REFERENCES PalBasics

# Multiplicity and Participation Constraints

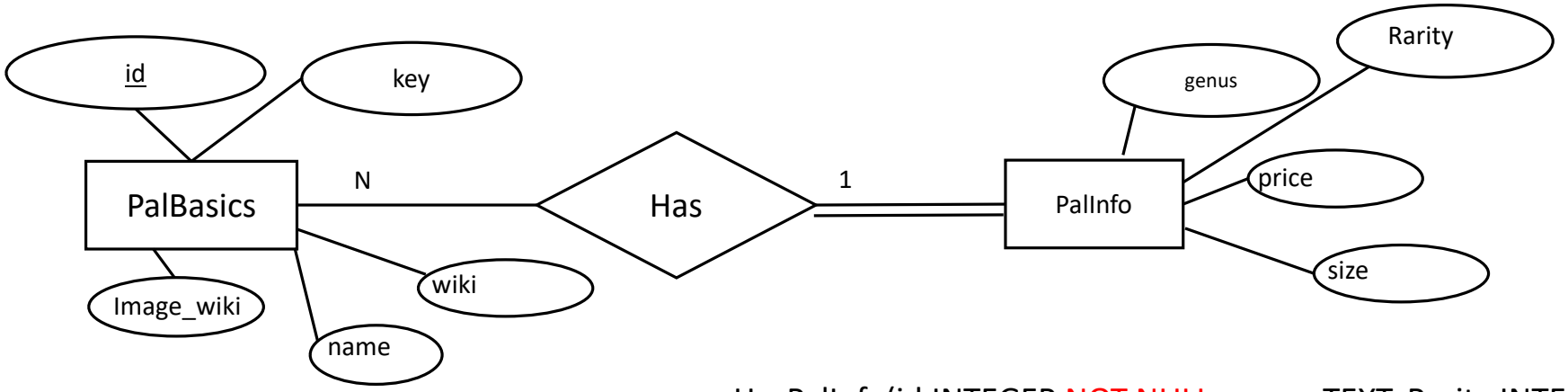

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




2) The imported attribute (id), cannot be NULL BECAUSE OF THE TOTAL PARTICIPATION.

# Multiplicity and Participation Constraints

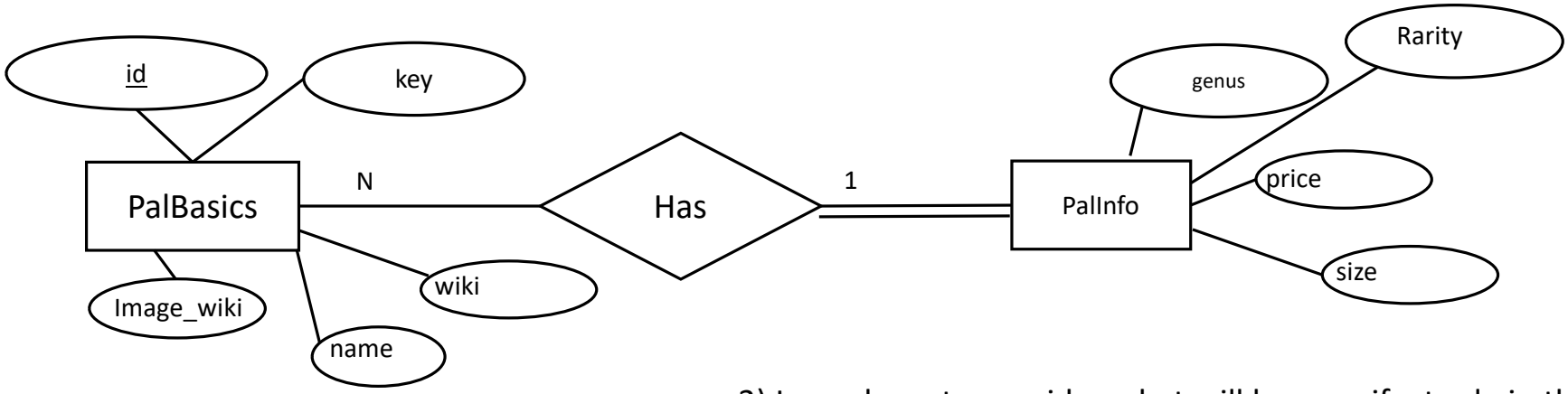

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




HasPalInfo(id INTEGER NOT NULL, genus TEXT, Rarity INTEGER,
price FLOAT, size INTEGER)
FOREIGN KEY id REFERENCES PalBasics

# Multiplicity and Participation Constraints


For a given ‘id’ value, there may be at most one PalInfo.
For a given PalInfo, there must be at least one “id” value.




3) I now have to consider what will happen if a tuple in the PalBasics is removed, options: ON DELETE CASCADE/REJECT

# Referential Integrity

Lancaster University

## PalBasics

| id | key | name | wiki | image_wiki |
|---|---|---|---|---|
| 1 | 1 | Lamball | https://palworld.fandom.com/wiki/Lamball | https://static.wikia.nocookie.net/palworld/images/0/01/Lamball_menu.png/ |
| 2 | 2 | Cattiva | https://palworld.fandom.com/wiki/Cattiva | https://static.wikia.nocookie.net/palworld/images/5/51/Cattiva_menu.png/ |
| 3 | 3 | Chikipi | https://palworld.fandom.com/wiki/Chikipi | https://static.wikia.nocookie.net/palworld/images/f/f4/Chikipi_menu.png/ |
| 4 | 4 | Lifmunk | https://palworld.fandom.com/wiki/Lifmunk | https://static.wikia.nocookie.net/palworld/images/d/dc/Lifmunk_menu.png/ |
| 5 | 5 | Foxparks | https://palworld.fandom.com/wiki/Foxparks | https://static.wikia.nocookie.net/palworld/images/d/d7/Foxparks_menu.png/ |
| 6 | 6 | Fuack | https://palworld.fandom.com/wiki/Fuack | https://static.wikia.nocookie.net/palworld/images/5/5e/Fuack_menu.png/ |
| 7 | 7 | Sparkit | https://palworld.fandom.com/wiki/Sparkit | https://static.wikia.nocookie.net/palworld/images/c/ce/Sparkit_menu.png/ |
| 8 | 8 | Tanzee | https://palworld.fandom.com/wiki/Tanzee | https://static.wikia.nocookie.net/palworld/images/4/40/Tanzee_menu.png/ |
| 9 | 9 | Rooby | https://palworld.fandom.com/wiki/Rooby | https://static.wikia.nocookie.net/palworld/images/2/21/Rooby_menu.png/ |
| 10 | 10 | Pengullet | https://palworld.fandom.com/wiki/Pengullet | https://static.wikia.nocookie.net/palworld/images/3/38/Pengullet_menu.png/ |
| 11 | 11 | Penking | https://palworld.fandom.com/wiki/Penking | https://static.wikia.nocookie.net/palworld/images/b/b5/Penking_menu.png/ |
| 12 | 12 | Jolthog | https://palworld.fandom.com/wiki/Jolthog | https://static.wikia.nocookie.net/palworld/images/5/52/Jolthog_menu.png/ |
| 13 | 13 | Gumoss | https://palworld.fandom.com/wiki/Gumoss | https://static.wikia.nocookie.net/palworld/images/2/2e/Gumoss_menu.png/ |
| 14 | 14 | Vixy | https://palworld.fandom.com/wiki/Vixy | https://static.wikia.nocookie.net/palworld/images/9/9e/Vixy_menu.png/ |
| 15 | 15 | Hoocrates | https://palworld.fandom.com/wiki/Hoocrates | https://static.wikia.nocookie.net/palworld/images/e/ef/Hoocrates_menu.png/ |

## PalInfo

| id | genus | rarity | price | size |
|---|---|---|---|---|
| 1 | humanoid | 1 | 1000 | xs |
| 2 | humanoid | 1 | 1000 | xs |
| 3 | bird | 1 | 1000 | xs |
| 4 | humanoid | 1 | 1010 | xs |
| 5 | fourlegged | 1 | 1040 | xs |
| 6 | humanoid | 1 | 1120 | xs |
| 7 | humanoid | 1 | 1030 | xs |
| 8 | humanoid | 1 | 1280 | xs |
| 9 | fourlegged | 2 | 2950 | s |
| 10 | humanoid | 1 | 1080 | xs |
| 11 | humanoid | 6 | 5410 | l |
| 12 | fourlegged | 1 | 1060 | xs |
| 13 | other | 1 | 1310 | xs |
| 14 | fourlegged | 2 | 1000 | xs |
| 15 | bird | 1 | 1050 | xs |
| 16 | fourlegged | 1 | 1000 | m |
| 17 | humanoid | 1 | 1050 | xs |
| 18 | fourlegged | 1 | 1420 | xs |
| 19 | humanoid | 1 | 1330 | xs |
| 20 | fourlegged | 1 | 1680 | s |

# Referential Integrity

Lancaster University

PalBasics

| id | key | name | wiki | image_wiki |
|---|---|---|---|---|
| 2 | 2 | Cattiva | https://palworld.fandom.com/wiki/Cattiva | https://static.wikia.nocookie.net/palworld/images/5/51/Cattiva_menu.png/ |
| 3 | 3 | Chikipi | https://palworld.fandom.com/wiki/Chikipi | https://static.wikia.nocookie.net/palworld/images/f/f4/Chikipi_menu.png/ |
| 4 | 4 | Lifmunk | https://palworld.fandom.com/wiki/Lifmunk | https://static.wikia.nocookie.net/palworld/images/d/dc/Lifmunk_menu.png/ |
| 5 | 5 | Foxparks | https://palworld.fandom.com/wiki/Foxparks | https://static.wikia.nocookie.net/palworld/images/d/d7/Foxparks_menu.png/ |
| 6 | 6 | Fuack | https://palworld.fandom.com/wiki/Fuack | https://static.wikia.nocookie.net/palworld/images/5/5e/Fuack_menu.png/ |
| 7 | 7 | Sparkit | https://palworld.fandom.com/wiki/Sparkit | https://static.wikia.nocookie.net/palworld/images/c/ce/Sparkit_menu.png/ |
| 8 | 8 | Tanzee | https://palworld.fandom.com/wiki/Tanzee | https://static.wikia.nocookie.net/palworld/images/4/40/Tanzee_menu.png/ |
| 9 | 9 | Rooby | https://palworld.fandom.com/wiki/Rooby | https://static.wikia.nocookie.net/palworld/images/2/21/Rooby_menu.png/ |
| 10 | 10 | Pengullet | https://palworld.fandom.com/wiki/Pengullet | https://static.wikia.nocookie.net/palworld/images/3/38/Pengullet_menu.png/ |
| 11 | 11 | Penking | https://palworld.fandom.com/wiki/Penking | https://static.wikia.nocookie.net/palworld/images/b/b5/Penking_menu.png/ |
| 12 | 12 | Jolthog | https://palworld.fandom.com/wiki/Jolthog | https://static.wikia.nocookie.net/palworld/images/5/52/Jolthog_menu.png/ |
| 13 | 13 | Gumoss | https://palworld.fandom.com/wiki/Gumoss | https://static.wikia.nocookie.net/palworld/images/2/2e/Gumoss_menu.png/ |
| 14 | 14 | Vixy | https://palworld.fandom.com/wiki/Vixy | https://static.wikia.nocookie.net/palworld/images/9/9e/Vixy_menu.png/ |
| 15 | 15 | Hoocrates | https://palworld.fandom.com/wiki/Hoocrates | https://static.wikia.nocookie.net/palworld/images/e/ef/Hoocrates_menu.png/ |

PalInfo

| id | genus | rarity | price | size |
|---|---|---|---|---|
| 1 | humanoid | 1 | 1000 | xs |
| 2 | humanoid | 1 | 1000 | xs |
| 3 | bird | 1 | 1000 | xs |
| 4 | humanoid | 1 | 1010 | xs |
| 5 | fourlegged | 1 | 1040 | xs |
| 6 | humanoid | 1 | 1120 | xs |
| 7 | humanoid | 1 | 1030 | xs |
| 8 | humanoid | 1 | 1280 | xs |
| 9 | fourlegged | 2 | 2950 | s |
| 10 | humanoid | 1 | 1080 | xs |
| 11 | humanoid | 6 | 5410 | l |
| 12 | fourlegged | 1 | 1060 | xs |
| 13 | other | 1 | 1310 | xs |
| 14 | fourlegged | 2 | 1000 | xs |
| 15 | bird | 1 | 1050 | xs |
| 16 | fourlegged | 1 | 1000 | m |
| 17 | humanoid | 1 | 1050 | xs |
| 18 | fourlegged | 1 | 1420 | xs |
| 19 | humanoid | 1 | 1330 | xs |
| 20 | fourlegged | 1 | 1680 | s |

# Referential Integrity

Lancaster University

PalBasics

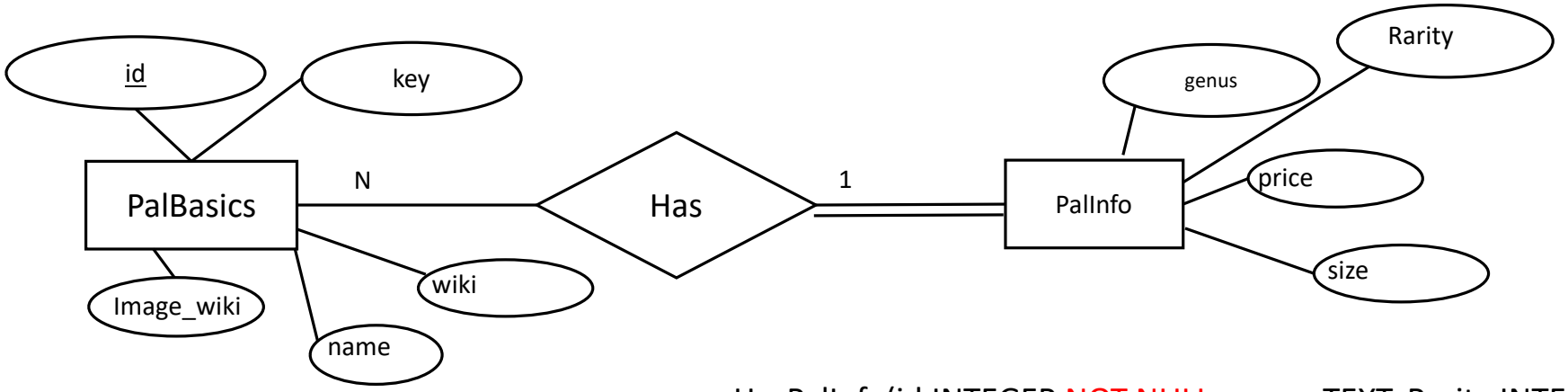
| id | key | name | wiki | image_wiki |
|---|---|---|---|---|
| 2 | 2 | Cattiva | https://palworld.fandom.com/wiki/Cattiva | https://static.wikia.nocookie.net/palworld/images/5/51/Cattiva_menu.png/ |
| 3 | 3 | Chikipi | https://palworld.fandom.com/wiki/Chikipi | https://static.wikia.nocookie.net/palworld/images/f/f4/Chikipi_menu.png/ |
| 4 | 4 | Lifmunk | https://palworld.fandom.com/wiki/Lifmunk | https://static.wikia.nocookie.net/palworld/images/d/dc/Lifmunk_menu.png/ |
| 5 | 5 | Foxparks | https://palworld.fandom.com/wiki/Foxparks | https://static.wikia.nocookie.net/palworld/images/d/d7/Foxparks_menu.png/ |
| 6 | 6 | Fuack | https://palworld.fandom.com/wiki/Fuack | https://static.wikia.nocookie.net/palworld/images/5/5e/Fuack_menu.png/ |
| 7 | 7 | Sparkit | https://palworld.fandom.com/wiki/Sparkit | https://static.wikia.nocookie.net/palworld/images/c/ce/Sparkit_menu.png/ |
| 8 | 8 | Tanzee | https://palworld.fandom.com/wiki/Tanzee | https://static.wikia.nocookie.net/palworld/images/4/40/Tanzee_menu.png/ |
| 9 | 9 | Rooby | https://palworld.fandom.com/wiki/Rooby | https://static.wikia.nocookie.net/palworld/images/2/21/Rooby_menu.png/ |
| 10 | 10 | Pengullet | https://palworld.fandom.com/wiki/Pengullet | https://static.wikia.nocookie.net/palworld/images/3/38/Pengullet_menu.png/ |
| 11 | 11 | Penking | https://palworld.fandom.com/wiki/Penking | https://static.wikia.nocookie.net/palworld/images/b/b5/Penking_menu.png/ |
| 12 | 12 | Jolthog | https://palworld.fandom.com/wiki/Jolthog | https://static.wikia.nocookie.net/palworld/images/5/52/Jolthog_menu.png/ |
| 13 | 13 | Gumoss | https://palworld.fandom.com/wiki/Gumoss | https://static.wikia.nocookie.net/palworld/images/2/2e/Gumoss_menu.png/ |
| 14 | 14 | Vixy | https://palworld.fandom.com/wiki/Vixy | https://static.wikia.nocookie.net/palworld/images/9/9e/Vixy_menu.png/ |
| 15 | 15 | Hoocrates | https://palworld.fandom.com/wiki/Hoocrates | https://static.wikia.nocookie.net/palworld/images/e/ef/Hoocrates_menu.png/ |

PalInfo

| id | genus | rarity | price | size |
|---|---|---|---|---|
| 1 | humanoid | 1 | 1000 | xs |
| 2 | humanoid | 1 | 1000 | xs |
| 3 | bird | 1 | 1000 | xs |
| 4 | humanoid | 1 | 1010 | xs |
| 5 | fourlegged | 1 | 1040 | xs |
| 6 | humanoid | 1 | 1120 | xs |
| 7 | humanoid | 1 | 1030 | xs |
| 8 | humanoid | 1 | 1280 | xs |
| 9 | fourlegged | 2 | 2950 | s |
| 10 | humanoid | 1 | 1080 | xs |
| 11 | humanoid | 6 | 5410 | l |
| 12 | fourlegged | 1 | 1060 | xs |
| 13 | other | 1 | 1310 | xs |
| 14 | fourlegged | 2 | 1000 | xs |
| 15 | bird | 1 | 1050 | xs |
| 16 | fourlegged | 1 | 1000 | m |
| 17 | humanoid | 1 | 1050 | xs |
| 18 | fourlegged | 1 | 1420 | xs |
| 19 | humanoid | 1 | 1330 | xs |
| 20 | fourlegged | 1 | 1680 | s |

# Multiplicity and Participation Constraints

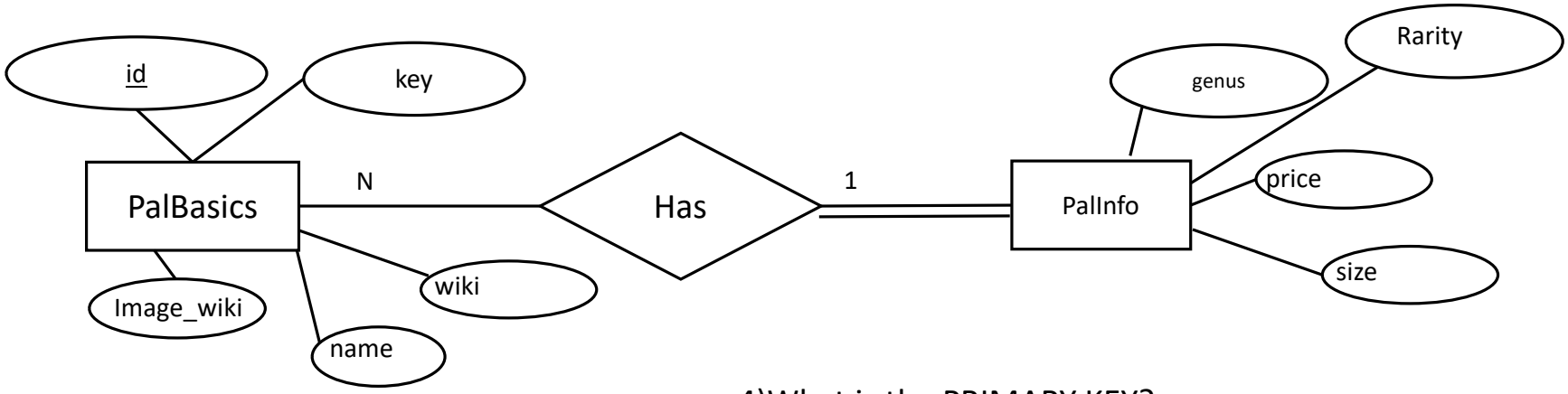

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




HasPalInfo(id INTEGER NOT NULL, genus TEXT, Rarity INTEGER, price FLOAT, size INTEGER)
FOREIGN KEY id REFERENCES PalBasics ON DELETE REJECT

# Multiplicity and Participation Constraints

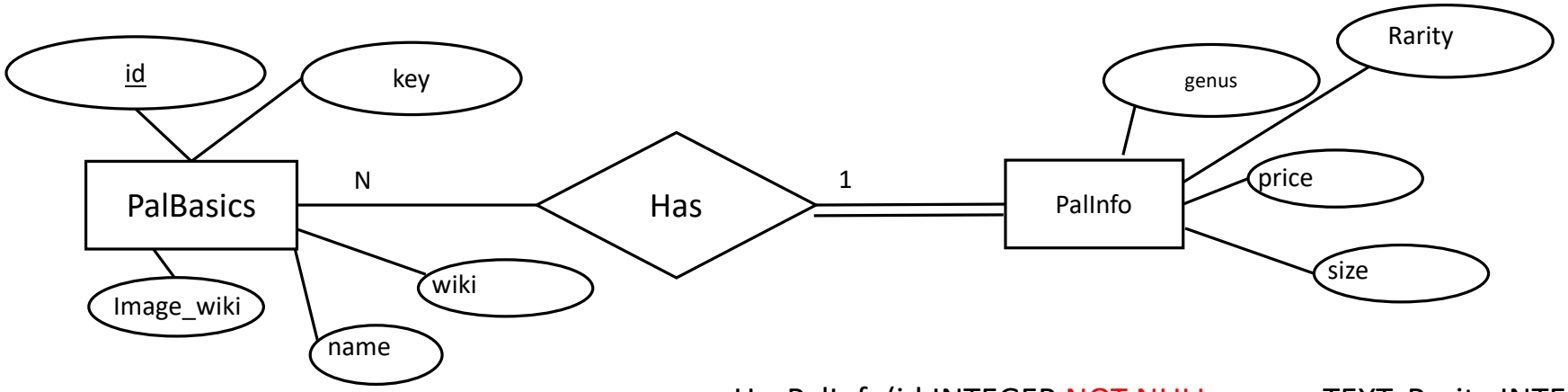

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




4)What is the PRIMARY KEY?

# Multiplicity and Participation Constraints

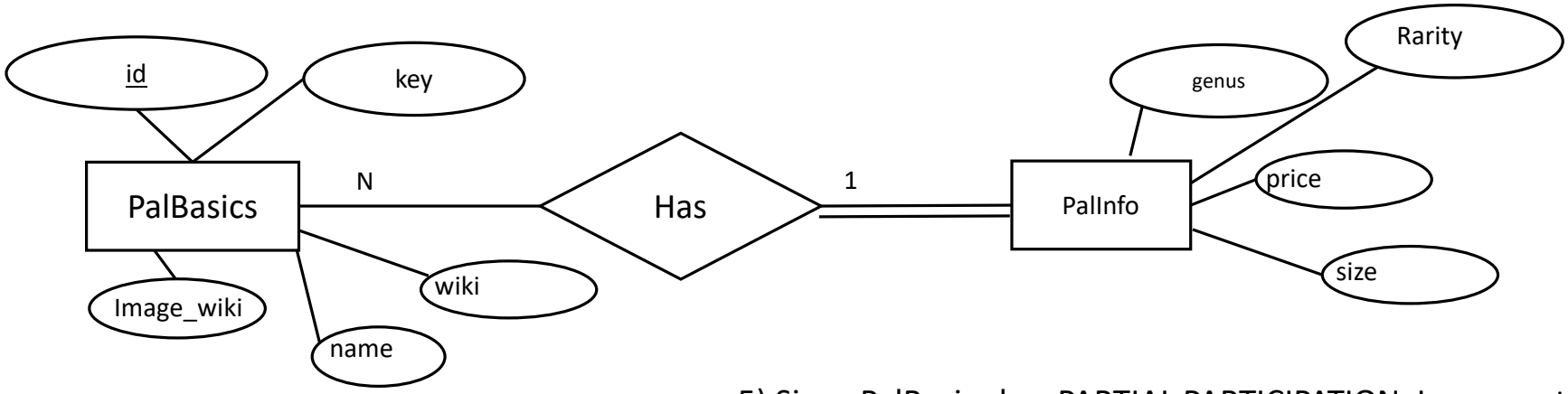

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




HasPalInfo(id INTEGER NOT NULL, genus TEXT, Rarity INTEGER, price FLOAT, size INTEGER)
FOREIGN KEY id REFERENCES PalBasics ON DELETE REJECT.
PRIMARY KEY id.

# Multiplicity and Participation Constraints

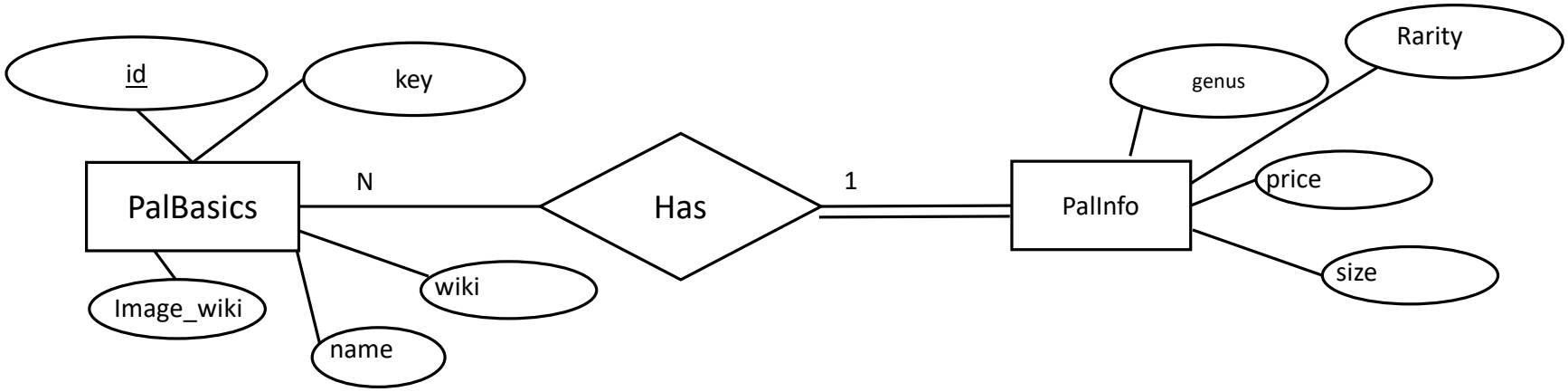

For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




5) Since PalBasics has PARTIAL PARTICIPATION, I can create it as an independent table.

# Multiplicity and Participation Constraints


For a given tuple in PalBasics, there may be at most one PalInfo tuple.
For a given PalInfo tuple, there must be at least one tuple in PalBasics.




RS: *Palbasics*(id INTEGER, key INTEGER, name TEXT, wiki TEXT, image_wiki TEXT)
PK:key

# Revision

Lancaster University

- Week 3
  - Functional Dependencies and Normal Forms
  - Functional dependency
    - If for a given attribute X, there are tuples that agree on a value, then the same tuples must also agree on a value of attribute Y. Hence, X->Y.
  - Not all FDs are trivial, we need to use Armstrong's Axioms to find others.
  - We have seen 1NF, 2NF, 3NF and BCNF

# Functional Dependencies (FORMAL)

Lancaster University


- A functional dependency between attributes X, Y ($X \longrightarrow Y$) holds over relation R if, for every allowable instance *r* of R:
  - $t1 \in r,\ t2 \in r,\ such\ that\ \prod_X (t1) = \prod_X (t2)$ implies $\prod_Y (t1) = \prod_Y (t2)$
  - i.e., given two tuples in *r*, if the ID values agree, then the NAME values must also agree.

| | ID | NAME | MOTION |
|---|---|---|---|
| *t1* | allay | Arrow | 23 |
| | Arrow | MAGMA CUBE | 44 |
| | Arrow | MAGMA CUBE | 1 |
| *t2* | allay | Arrow | 54 |


- Consider this table.
- Can you observe relations between values of different attributes?

| AIR | CUSTOMNAME | CUSTOMNAMEVISIBLE | FALLDISTANCE | FIRE | GLOWING | ID | NAME | INVULNERABLE | MOTION |
|---|---|---|---|---|---|---|---|---|---|
| 5500 | “VERRDO” | 1 | 34 | 44 | 1 | allay | ALLAY | 0 | 45 |
| 234 | “Hero” | 0 | 22 | 12 | 0 | Magma_cube | MAGMA CUBE | 0 | 12 |
| 12 | “Zappara” | 1 | 22 | 33 | 0 | arrow | ARROW | 1 | 22 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 12 |
| 215 | “uraz” | 0 | 21 | 562 | 0 | Magma_cube | MAGMA CUBE | 0 | 88 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 98 |


- Consider this table.
- Can you observe relations between values of different attributes?

| AIR | CUSTOMNAME | CUSTOMNAMEVISIBLE | FALLDISTANCE | FIRE | GLOWING | ID | NAME | INVULNERABLE | MOTION |
|---|---|---|---|---|---|---|---|---|---|
| 5500 | “VERRDO” | 1 | 34 | 44 | 1 | allay | ALLAY | 0 | 45 |
| 234 | “Hero” | 0 | 22 | 12 | 0 | Magma_cube | MAGMA CUBE | 0 | 12 |
| 12 | “Zappara” | 1 | 22 | 33 | 0 | arrow | ARROW | 1 | 22 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 12 |
| 215 | “uraz” | 0 | 21 | 562 | 0 | Magma_cube | MAGMA CUBE | 0 | 88 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 98 |


- Since ID->NAME (or lets say I->N for short)
- Then I can decompose the table into two
- Where I keep the main table without N and create another table having I,N.

| AIR | CUSTOMNAME | CUSTOMNAMEVISIBLE | FALLDISTANCE | FIRE | GLOWING | ID | NAME | INVULNERABLE | MOTION |
|---|---|---|---|---|---|---|---|---|---|
| 5500 | “VERRDO” | 1 | 34 | 44 | 1 | allay | ALLAY | 0 | 45 |
| 234 | “Hero” | 0 | 22 | 12 | 0 | Magma_cube | MAGMA CUBE | 0 | 12 |
| 12 | “Zappara” | 1 | 22 | 33 | 0 | arrow | ARROW | 1 | 22 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 12 |
| 215 | “uraz” | 0 | 21 | 562 | 0 | Magma_cube | MAGMA CUBE | 0 | 88 |
| 0 | “Wirdo” | 1 | -20 | 67 | 0 | Boat_chest | BOAT WITH CHEST | 0 | 98 |

## DECOMPOSITION

| AIR | CUSTOMNAME | CUSTOMNAMEVISIBLE | FALLDISTANCE | FIRE | GLOWING | ID | INVULNERABLE | MOTION |
|---|---|---|---|---|---|---|---|---|
| 5500 | “VERRDO” | 1 | 34 | 44 | 1 | allay | 0 | 45 |
| 234 | “Hero” | 0 | 22 | 12 | 0 | Magma_cube | 0 | 12 |
| 12 | “Zappara” | 1 | 22 | 33 | 0 | arrow | 1 | 22 |
| 0 | “owned” | 1 | -20 | -20 | 0 | Boat_chest | 0 | 12 |
| 215 | “uraz” | 0 | 21 | 562 | 0 | Magma_cube | 0 | 88 |
| 0 | “Wirdo” | 1 | -20 | 67 | 0 | Boat_chest | 0 | 98 |

| ID | NAME |
|---|---|
| allay | ALLAY |
| Magma_cube | MAGMA CUBE |
| arrow | ARROW |
| Boat_chest | BOAT WITH CHEST |

# Reasoning About FDs

Lancaster University

- $F^{+}$ *closure of F* is the set of all FDs that are implied by *F*.
- Armstrong's Axioms (X, Y, Z are sets of attributes):
  - *Reflexivity*: If X $\in$ Y, then Y -> X (a trivial FD) "XY->Y"
  - *Augmentation*: If X -> Y, then XZ ->YZ for any Z
  - *Transitivity*: If X -> Y and Y->Z, then X->Z
  - *Union*: If X -> Y and X -> Z, then X -> YZ
  - *Decomposition*: If X -> YZ, then X -> Y and X -> Z

- These are *sound* and *complete* inference rules for FDs!

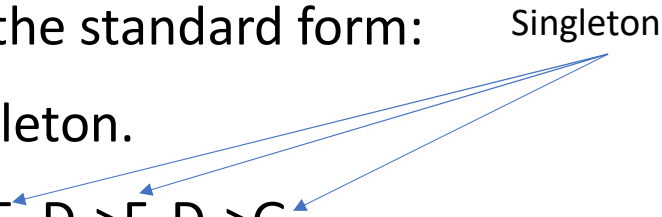
# Revealing hidden FDs

Lancaster University

Rule 1: start by representing the FDs in the standard form:

The implied attribute list is a singleton.

So instead of D->EFG, create D->E, D->F, D->G.




Rule 2: By using the axioms, try to reveal implied FDs.

Rule 3: Use Augmentation to cover the remaining attributes.

# Normal Forms Contd.

Lancaster University

- Ex: R = ABCDE, F= { BC->D, AC->BE, B->E , AC->B, AC->E,AC->BC,AC->D }
- AC->BE……..AC->B, AC->E
- ACC->BC…AC->BC
- BC->D….AC->BC->D…AC->D
- AC is the candidate key as AC->BE, AC->D,
- AC→ABCDE.

# Prime and Non-prime attributes

Lancaster University

- Prime attribute: any attribute that is part of a candidate key.
- Non-prime attributes: The rest of the attributes.
- First Normal Form: No set-valued attributes (only atomic values).
- Second Normal Form: Relation must be in 1st NF, and a part of the PK cannot imply a non-prime attribute. This rule is valid if the PK is composite, i.e., having more than one attribute. If the table is in 1st NF and has a singleton PK, it is in 2nd NF.
- Third Normal Form: Relation must be in 2nd NF, and a non-prime attribute cannot imply another non-prime attribute.
- Boyce-Codd Normal Form: Relation must be in 3rd NF and there exists only trivial FDs and FDs where implying attributes constitute a superkey for the relation.

# Prime and Non-prime attributes

Lancaster University

- Prime attribute: any attribute that is part of a candidate key.
- Non-prime attributes: The rest of the attributes.
- First Normal Form: No set-valued attributes (only atomic values).
- Second Normal Form: Relation must be in 1st NF, and a part of the PK cannot imply a non-prime attribute. This rule is valid if the PK is composite, i.e., having more than one attribute. If the table is in 1st NF and has a singleton PK, it is in 2nd NF.
- Third Normal Form: Relation must be in 2nd NF, and a non-prime attribute cannot imply another non-prime attribute.
- Boyce-Codd Normal Form: Relation must be in 3rd NF and there exists only trivial FDs and FDs where implying attributes constitute a superkey for the relation.

# Prime and Non-prime attributes

Lancaster University

- Prime attribute: any attribute that is part of a candidate key.
- Non-prime attributes: The rest of the attributes.
- First Normal Form: No set-valued attributes (only atomic values).
- Second Normal Form: Relation must be in 1st NF, and a part of the PK cannot imply a non-prime attribute. This rule is valid if the PK is composite, i.e., having more than one attribute. If the table is in 1st NF and has a singleton PK, it is in 2nd NF.
- Third Normal Form: Relation must be in 2nd NF, and a non-prime attribute cannot imply another non-prime attribute.
- Boyce-Codd Normal Form: Relation must be in 3rd NF and there exists only trivial FDs and FDs where implying attributes constitute a superkey for the relation.

# Prime and Non-prime attributes

Lancaster University

- Prime attribute: any attribute that is part of a candidate key.
- Non-prime attributes: The rest of the attributes.
- First Normal Form: No set-valued attributes (only atomic values).
- Second Normal Form: Relation must be in 1st NF, and a part of the PK cannot imply a non-prime attribute. This rule is valid if the PK is composite, i.e., having more than one attribute. If the table is in 1st NF and has a singleton PK, it is in 2nd NF.
- Third Normal Form: Relation must be in 2nd NF, and a non-prime attribute cannot imply another non-prime attribute.
- Boyce-Codd Normal Form: Relation must be in 3rd NF and there exists only trivial FDs and FDs where implying attributes constitute a superkey for the relation.

# Prime and Non-prime attributes

Lancaster University

- Prime attribute: any attribute that is part of a candidate key.
- Non-prime attributes: The rest of the attributes.
- First Normal Form: No set-valued attributes (only atomic values).
- Second Normal Form: Relation must be in 1st NF, and a part of the PK cannot imply a non-prime attribute. This rule is valid if the PK is composite, i.e., having more than one attribute. If the table is in 1st NF and has a singleton PK, it is in 2nd NF.
- Third Normal Form: Relation must be in 2nd NF, and a non-prime attribute cannot imply another non-prime attribute.
- Boyce-Codd Normal Form: Relation must be in 3rd NF, and there exists only (1) trivial FDs and (2) FDs where implying attributes constitute a superkey for the relation.

# Revision

Lancaster University

- Week 4
  - Relational Algebra
  - We have seen set of operators to create mathematical formulations for queries.
    - Selection
    - Projection
    - Cross Product
    - Theta Join, Equi Join, Natural Join, Set Difference, Division, Union and Intersection

# Relational Algebra

Lancaster University

- A mathematical foundation for querying relational databases.
- Contains several operators that receive relations and return relations.
  - Selection $\sigma$ — Returns the required tuples
  - Projection $\pi$ — Returns the required attributes
  - Cross-product $\times$ — Returns the cartesian product of two relations
  - Union $\cup$ — Returns the union of two relations
  - Intersection $\cap$ — Returns the intersection of two relations
  - Division / — Returns fully associated tuples
  - Join $\bowtie$ — Returns rows obeying the Join-Condition after performing a cross-product.
  - Renaming $\rho$ — Renames the attributes based on their indices, which start from ‘1’.
- Each operator has its own syntax.

# Revision

Lancaster University

- Week 5
  - MySQL
  - Industry standard scripting language for relational databases.
  - Conceptual Evaluation Strategy for Select implies that we need to cross-product the relations in relation list and omit the rows that cannot satisfy conditions given with the WHERE clause.

# MySQL

Lancaster University

- An industry standard query and database scripting language.
- We have seen DDL scripts:
  - CREATE DATABASE,
  - CREATE TABLE, DROP TABLE,
- We have seen DML scripts:
  - INSERT INTO..VALUES
  - SELECT..FROM..WHERE,
  - Nested queries, Nested Queries with correlation,
- EXISTS, ALL, ANY, UNIQUE, EXCEPT, INTERSECTION, UNION.

# EXAM QUESTIONS

Consider the tables:

**Players (playerId, AccountNo, email)**

**Characters (AccountNo, CharName, Power, Rating, Money, ExperienceScore, Item_type)**

**Inventory (Item_type, Price, Wearable)**

3.a Write the SQL statements for the given queries:

A. Find the ID of a player who uses all items. **[3 marks]**

```
SELECT R0.playerId
FROM Players R0
WHERE R0.AccountNo IN ( SELECT R1.AcccountNo
                        FROM Characters R1
                        WHERE NOT EXISTS
                        (SELECT R2.Item_type FROM Inventory R2)
                        EXCEPT
                        (
                        SELECT R3.Item_type FROM INVENTORY R3
                        WHERE  R3.Item_type=R1.Item_type
                        )
                      );
```

Division to find all AccountNo's from Characters that use all items.

B. Find the Account number of a player having playerId="20" for each rating with at least 5 characters. **[5 marks]**

```
SELECT P.AccountNo, C.rating
FROM Players P,Characters C WHERE playerId="20"
GROUPBY C.rating
Having COUNT(*)>4;
```

C. Find the emails for the player whose id begins with "007". **[3 marks]**

```
SELECT email FROM Players
WHERE playerId LIKE '007%';
```

D. Find the email for the player whose id is "007". **[4 marks]**

```
SELECT email FROM Players
WHERE playerId = '007';
```