COIT20247 - Database Design & Development
Tutorial Solutions – Logical design (ER Transformation)

Note: due to the difficulty of formatting and displaying arrows in a word processor document, the "FOREIGN KEY" syntax will be used in this document. You may use the arrow syntax if you prefer.

1. **Students and courses:**

   STUDENT (<u>StudentID</u>, Name, Address)

   COURSE (<u>CourseID</u>, CourseName, CreditPoints)

   ENROLMENT (*<u>CourseID</u>*, *<u>StudentID</u>*)
      FOREIGN KEY (CourseID) REFERENCES Course
      FOREIGN KEY (StudentID) REFERENCES Student

   Note: you may have given ENROLMENT a different name – provided it is a sensible name, then it's acceptable.

2. **Buildings and offices:**

   BUILDING (<u>BuildingNbr</u>, BuildingName)

   OFFICE (*<u>BuildingNbr</u>*, <u>OfficeNbr</u>, OfficePhone)
      FOREIGN KEY (BuildingNbr) REFERENCES Building

3. **Employee supervision:**

   EMPLOYEE (<u>EmployeeID</u>, EmpName)

   SUPERVISION (*<u>SupervisorID</u>*, *<u>EmployeeID</u>*)
      FOREIGN KEY (SupervisorID) REFERENCES Employee
      FOREIGN KEY (EmployeeID) REFERENCES Employee

   Note: You might have named the SUPERVISION relation differently or named the attributes within it different. Provided the names are sensible, this is fine.

4. **Staff and departments:**

   STAFF (<u>StaffID</u>, StaffName, *DeptID*)
      FOREIGN KEY (DeptID) REFERENCES Department

   DEPARTMENT (<u>DeptID</u>, DeptName, ContactPh, *HeadID*)
      FOREIGN KEY (HeadID) REFERENCES Staff

   Note: The attribute name *HeadID* was used in the DEPARTMENT relation above as it is more meaningful than *StaffID*. It also demonstrates that the name of the foreign key *does not have to match the name of the corresponding primary key*. That is why the arrow/foreign key statement is important – it is not always going to be obvious which relation the foreign key refers to. It is also perfectly acceptable to use *StaffID* in place of *HeadID*.

5. **Students, courses & campuses:**

CAMPUS (CampusID, CampusName)

COURSE (CourseID, CourseName)

STUDENT (StudentID, Name, DateOfBirth, Street, City, State, PostCode, StudyLevel, StudentType)

ENROLMENT (StudentID, CourseID)
FOREIGN KEY (StudentID) REFERENCES Student
FOREIGN KEY (CourseID) REFERENCES Course

STUDENT-PHONE (*StudentID*, Number, Type)
FOREIGN KEY (StudentID) REFERENCES Student

STUDENT-EMAIL (*StudentID*, Email)
FOREIGN KEY (StudentID) REFERENCES Student

FLEX (*StudentID*, StudyGroup)
FOREIGN KEY (StudentID) REFERENCES Student

ON-CAMPUS (*StudentID*, GymMember, *CampusID*)
FOREIGN KEY (StudentID) REFERENCES Student
FOREIGN KEY (CampusID) REFERENCES Campus

NON-AWARD (*StudentID*, Employer-School)
FOREIGN KEY (StudentID) REFERENCES Student

UNDERGRAD (*StudentID*, GovtSupported)
FOREIGN KEY (StudentID) REFERENCES Student

Notes:

- You may have chosen to include Age in the STUDENT relation. This is a valid design choice, but you would have to be able to justify your decision. It has been left out of the above design as it is a very straightforward matter to calculate the age of a student based upon the DateOfBirth attribute. Age would rarely be needed, and easily calculated when required. By contrast, if Age were stored, it would become out-of-date annually and would need to be checked before being used anyway.
- (StudentID, Type) would be accepted as a primary key for the STUDENT-PHONE relation, although (StudentID, Number) is arguably a better choice. However, note that (StudentID) on its own **is not sufficient for a primary key** and would be marked as wrong!

6. **Best books:** Left as an exercise.

7. **Real estate:** Left as an exercise.

8. **Milk dairy:**

SUPPLIER (SupplierID, SupplierName)

FARMER-SUPPLIER (*FarmerSupplierD*, *SocietySupplierID*)
FOREIGN KEY (FarmerSupplierID) REFERENCES Supplier
FOREIGN KEY (SocietySupplierID) REFERENCES Society-Supplier

SOCIETY-SUPPLIER (*SocietySupplierD*, PresidentName, PresidentPhone, SecretaryName, SecretaryPhone)
FOREIGN KEY (SocietySupplierID) REFERENCES Supplier

CHILLING-CENTRE-SUPPLIER (*ChillingCentreSupplierD*, ManagerName)
FOREIGN KEY (ChillingCentreSupplierID) REFERENCES Supplier

CHILLING-CENTRE-PHONE (*ChillingCentreSupplierD*, ContactPhone)
FOREIGN KEY (ChillingCentreSupplierID) REFERENCES Chilling-Centre-Supplier

BATCH (BatchID, DateSupplied, QtySupplied, *SupplierID, PaymentID*)
FOREIGN KEY (SupplierID) REFERENCES Supplier
FOREIGN KEY (BatchID) REFERENCES Batch

SAMPLE (SampleID, Timestamp, Fat%, SNF%, *BatchID*)
FOREIGN KEY (BatchID) REFERENCES Batch

PAYMENT (PaymentID, DatePaid, AmtPaid, *SupplierID*)
FOREIGN KEY (SupplierID) REFERENCES Supplier

9. **Quikfix electronics:**

CUSTOMER (CustomerID, CustomerName, Street, Suburb, State, Postcode)

CUSTOMER-PHONE (*CustomerID*, ContactPhone)
FOREIGN KEY (CustomerID) REFERENCES Customer

CUSTOMER (CustomerID, CustomerName, Street, Suburb, State, Postcode)

ITEM-TYPE (ItemTypeID, ItemType)

ITEM (ItemID, Description, Make, Model, SerialNbr, *CustomerID, ItemTypeID*)
FOREIGN KEY (CustomerID) REFERENCES Customer
FOREIGN KEY (ItemTypeID) REFERENCES ItemType

EMPLOYEE (EmployeeID, EmployeeName, Street, Suburb, State, Postcode, ContactPhone)

REPAIR (JobNbr, ProblemDescription, NbrHours, TotalAmtDue, Completed, *ItemID, EmployeeID*)
FOREIGN KEY (ItemID) REFERENCES Item
FOREIGN KEY (EmployeeID) REFERENCES Employee

PART-TYPE (PartTypeID, Description, PricePerUnit)

PARTS-USED (*JobNbr*, *PartTypeID*)
FOREIGN KEY (JobNbr) REFERENCES Repair
FOREIGN KEY (PartTypeID) REFERENCES Part-Type

PAYMENT (PaymentID, DatePaid, AmtPaid, *JobNbr*)
FOREIGN KEY (JobNbr) REFERENCES Repair

Note: It is possible to put PaymentID into Repair (instead of the other way around) and would be accepted as a "valid" solution; however the solution above (placing JobNbr into Payment) reduces the number of nulls in the database.

10. **Building Brokerage Services:** Left as an exercise.