HABIB UNIVERSITY

Database Systems

CS/CE 355/373 Fall 2023

Instructor: Maria Samad

Normalization Solution

For the given relation schemas, get the 1NF, 2NF and 3NF where needed

- CustOrder (OrderID, CustomerID, (OrderDate), TotalAmount, CustomerName, Receipt)
- SupplierProduct (SupplierID, ProductID, SupplierName, (SupplierAddress), ProductName, MonthlyReport)
- StudentCourse (StudentID, StudentName, (CourseCode), CourseTitle, Semester, InstructorID)

ONE TENTATIVE SOLUTION:

- CustOrder (OrderID, CustomerID, (OrderDate), TotalAmount, CustomerName, Receipt)

1NF:

- First Normal Form schemas should contain all atomic attributes – here there are no multivalued attributes but composite attribute is there, i.e. Order Date, so we break it into OrderDay, OrderMonth, OrderYear and then redefine the schema to get the 1NF for it:
  - CustOrder (OrderID, CustomerID, OrderDay, OrderMonth, OrderYear, TotalAmount, CustomerName, Receipt)

2NF:

- For the Second Normal Form, we need to find all the functional dependencies of Primary/Candidate Keys with the non-prime attributes, and see if any of the composite ones are fully functional dependent, and which one has partial dependencies
- The functional dependencies for the 1NF schema are:
  - OrderID → OrderDay
  - OrderID → OrderMonth
  - OrderID → OrderYear
  - OrderID → TotalAmount
  - OrderID → Receipt
  - CustomerID → CustomerName
  - {OrderID, CustomerID} → Receipt
- The only composite key dependency is: {OrderID, CustomerID} → Receipt
- Check if this is fully dependent or partially dependent
- As we already have OrderID → Receipt, then this is partially dependent, cannot be decomposed into separate schema
- Therefore, we can decompose the non-2NF schema into two 2NF schemas:
  - Order (OrderID, CustomerID, OrderDay, OrderMonth, OrderYear, TotalAmount, Receipt)
    - Even though we didn't check the dependency between two primary keys, i.e. OrderID and CustomerID, but logically, CustomerID can be found out from OrderID, because we have assumed that Order IDs are always going to be unique
    - Here, we don't need to make CustomerID a primary key, as OrderID itself would be sufficient to get unique tuples, but CustomerID will be used as Foreign Key to get Customer Details if necessary
  - Customer (CustomerID, CustomerName)

3NF:

- For the Third-Normal Form, we need to look at the schemas from 2NF, and check if any of them is forming a transitive dependency
- Customer (CustomerID, CustomerName) is already in 3NF with only one non-prime attribute, so no transitivity possible here
- Check the other schema from 2NF, that is:
  Order (OrderID, CustomerID, OrderDay, OrderMonth, OrderYear, TotalAmount, Receipt)
- The transitive dependencies are:
  - OrderID → Receipt **AND** Receipt → CustomerID
  - OrderID → Receipt **AND** Receipt → OrderDay
  - OrderID → Receipt **AND** Receipt → OrderMonth
  - OrderID → Receipt **AND** Receipt → OrderYear
  - OrderID → Receipt **AND** Receipt → TotalAmount
- In these transitive dependencies, we need to check the two conditions of 3NF, i.e. for X → A:
  - X is a superkey
  - A is a prime attribute
- If any of the conditions are true then it is in 3NF
- OrderID is a superkey, so need to check for Receipt and it is fine for 3NF requirements
- What about the others?
- Receipt is a not a superkey, so check for its dependents:
  - None of the dependents, i.e. CustomerID, OrderDay, OrderMonth, OrderYear, TotalAmount, in this schema are prime attributes, so even the second condition is not satisfied. Hence this schema is not in 3NF
- To convert this into 3NF, we will decompose the schema and bring all the nonprime attributes dependencies into a separate schema, i.e.
  - Order (OrderID, Receipt)
  - OrderReceipt (Receipt, OrderDay, OrderMonth, OrderYear, TotalAmount, CustomerID)
    - ❖ This is not the final step, as we must check for any more transitive dependencies in the above schema
    - ❖ However, in this case there are no further transitive dependencies, hence this can be used as the final 3NF as it is
- Therefore, final 3NF schemas will be:
  - Order (OrderID, Receipt)
  - OrderReceipt (Receipt, OrderDay, OrderMonth, OrderYear, TotalAmount, CustomerID)
  - Customer (CustomerID, CustomerName)

- SupplierProduct (SupplierID, ProductID, SupplierName, (SupplierAddress), ProductName, MonthlyReport)

1NF:

- First Normal Form schemas should contain all atomic attributes – here there are no multivalued attributes but composite attribute is there, i.e. supplier address, so we break it into SupplierCity, SupplierPostalCode and then redefine the schema to get the 1NF for it:
  - SupplierProduct (SupplierID, ProductID, SupplierName, SupplierCity, SupplierPostalCode, ProductName, MonthlyReport)

2NF:

- For the Second Normal Form, we need to find all the functional dependencies of Primary/Candidate Keys with the non-prime attributes, and see if any of the composite ones are fully functional dependent, and which one has partial dependencies
- The functional dependencies for the 1NF schema are:
  - SupplierID → SupplierName
  - SupplierID → SupplierCity
  - SupplierID → SupplierPostalCode
  - ProductID → ProductName
  - {SupplierID, ProductID} → MonthlyReport
- The only composite key dependency is: {SupplierID, ProductID} → MonthlyReport
- Check if this is fully dependent or partially dependent
- As we do not have SupplierID → MonthlyReport, nor ProductID → MonthlyReport, then this is fully dependent, and should be represented as a separate schema
- Therefore, we can decompose the non-2NF schema into three 2NF schemas:
  - Report (SupplierID, ProductID, MonthlyReport)
  - Supplier (SupplierID, SupplierName, SupplierCity, SupplierPostalCode)
  - Product (ProductID, ProductName)

3NF:

- For the Third-Normal Form, we need to look at the schemas from 2NF, and check if any of them is forming a transitive dependency
- Product (ProductID, ProductName) is already in 3NF with only one non-prime attribute, so no transitivity here
- Report (SupplierID, ProductID, MonthlyReport) is also already in 3NF with only one non-prime attribute, so no transitivity here
- Check the remaining schema from 2NF, that is:
  Supplier (SupplierID, SupplierName, SupplierCity, SupplierPostalCode)
- The transitive dependencies are:
  - SupplierID → SupplierName **AND** SupplierName → SupplierCity
  - SupplierID → SupplierName **AND** SupplierName → SupplierPostalCode
- In these transitive dependencies, we need to check the two conditions of 3NF, i.e. for X → A:
  - X is a superkey
  - A is a prime attribute
- If any of the conditions are true then it is in 3NF
- SupplierID is a superkey, so need to check for SupplierName and it is fine for 3NF requirements
- What about the others?
- SupplierName is a not a superkey, so check for its dependents:
  - None of the dependents, i.e. SupplierCity, SupplierPostalCode, in this schema are prime attributes, so even the second condition is not satisfied. Hence this schema is not in 3NF
- To convert this into 3NF, we will decompose the schema and bring all the nonprime attributes dependencies into a separate schemas, i.e.
  - Supplier (SupplierID, SupplierName)
  - SupplierLocation (SupplierName, SupplierCity, SupplierPostalCode)
    - ❖ This is not the final step, as we must check for any more transitive dependencies in the above schema
    - ❖ However, in this case there are no further transitive dependencies, hence this can be used as the final 3NF as it is
- Therefore, final 3NF schemas will be:
  - Supplier (SupplierID, SupplierName)
  - SupplierLocation (SupplierName, SupplierCity, SupplierPostalCode)
  - Report (SupplierID, ProductID, MonthlyReport)
  - Product (ProductID, ProductName)

- StudentCourse (StudentID, StudentName, (CourseCode), CourseTitle, Semester, InstructorID)

The given schema is incomplete. As having StudentID being the primary key, we will never get unique tuples, so this schema should also have CourseCode as the primary key. Therefore the updated schema will be:

StudentCourse (StudentID, (CourseCode), StudentName, CourseTitle, Semester, InstructorID)

1NF:

- First Normal Form schemas should contain all atomic attributes – here there are no multivalued attributes but composite attribute is there, i.e. course code, so we break it into CourseDepartment, CouseNumber and then redefine the schema to get the 1NF for it:
  - StudentCourse (StudentID, CourseDepartment, CourseNumber, StudentName, CourseTitle, Semester, InstructorID)

2NF:

- For the Second Normal Form, we need to find all the functional dependencies of Primary/Candidate Keys with the non-prime attributes, and see if any of the composite ones are fully functional dependent, and which one has partial dependencies
- The functional dependencies for the 1NF schema are:
  - StudentID → StudentName
  - {CourseDepartment, CouseNumber} → CourseTitle
  - {StudentID, CourseDepartment, CouseNumber} → Semester
  - {StudentID, CourseDepartment, CouseNumber} → InstructorID
- For the composite keys, check if they are fully dependent or partially dependent.
- The composite keys are:
  - {CourseDepartment, CouseNumber} → CourseTitle
    - As we do not have CourseDepartment → CourseTitle, nor CouseNumber → CourseTitle, then this is fully dependent, and should be represented as a separate schema
  - {StudentID, CourseDepartment, CouseNumber} → Semester
    - Check for all the subsets and their dependencies:
      - StudentID → Semester ⟶ Not possible
      - CourseDepartment → Semester ⟶ Not possible
      - CouseNumber → Semester ⟶ Not possible
      - {StudentID, CourseDepartment}→ Semester ⟶ Not possible
      - {StudentID, CouseNumber}→ Semester ⟶ Not possible
      - {CourseDepartment, CouseNumber}→ Semester ⟶ Not possible
    - As neither is possible, so this is also fully dependent, and should be separated from the original schema
  - {StudentID, CourseDepartment, CouseNumber} → InstructorID
    - Check for all the subsets and their dependencies:
      - StudentID → InstructorID ⟶ Not possible
      - CourseDepartment → InstructorID ⟶ Not possible
      - CouseNumber → InstructorID ⟶ Not possible
      - {StudentID, CourseDepartment}→ InstructorID ⟶ Not possible
      - {StudentID, CouseNumber}→ InstructorID ⟶ Not possible
      - {CourseDepartment, CouseNumber}→ InstructorID ⟶ Not possible
    - As neither is possible, so this is also fully dependent, and should be separated from the original schema
- Therefore, we can decompose the non-2NF schema into three 2NF schemas:
  - Student (StudentID, StudentName)
  - Course (CourseDepartment, CourseNumber, CourseTitle)
  - CourseDetails (StudentID, CourseDepartment, CourseNumber, Semester, Instructor)

3NF:

- For the Third-Normal Form, we need to look at the schemas from 2NF, and check if any of them is forming a transitive dependency
- Student (StudentID, StudentName) is already in 3NF with only one non-prime attribute, so no transitivity here
- Course (CourseDepartment, CourseNumber, CourseTitle) is also already in 3NF with only one non-prime attribute, so no transitivity here
- Check the remaining schema from 2NF, that is:

  CourseDetails (StudentID, CourseDepartment, CourseNumber, Semester, Instructor)
- There are no transitive dependencies here, so this is also already in 3NF
- Therefore, final 3NF schemas will be the same as 2NF, i.e. :
  - Student (StudentID, StudentName)
  - Course (CourseDepartment, CourseNumber, CourseTitle)
  - CourseDetails (StudentID, CourseDepartment, CourseNumber, Semester, Instructor)