INFO90002 Week 6 Tutorial

Objectives

- Normalisation
- Learn SQL By Example

Section1 Normalisation

Normalization and normal forms

Normalization is a technique used to iteratively improve relations to remove undesired redundancy by decomposing relations and eliminating anomalies. The process is iterative and can be performed in stages generally referred to as Normal Forms. In First Normal Form (1NF), the relation is analysed and all repeating groups are identified to be decomposed into new relations. In Second Normal Form (2NF), all the partial dependencies are resolved/removed. The next stage is Third Normal Form (3NF) where all the transitive dependencies are removed.

Key Concepts:

Anomalies

Consider the following instance of the relation Allocation (CourseNumber, Tutor, Room, Seats):

CourseNumber	**Tutor**	**Room**	**Seats**
INFO20003	Farah	Alice Hoy 109	30
COMP10001	Farah	EDS 6	25
INFO30005	Patrick	Sidney Myer G09	20
COMP20005	Alan	Sidney Myer G09	20

An **update** anomaly is a data inconsistency that results from data redundancy and partial update when one or more instances of duplicated data are updated but not all. For example, suppose the room Sidney Myer G09 has been improved, and there are now 30 seats. In this single entity, we will have to update all other rows where room = Sidney Myer G09.

A **deletion** anomaly is an unintentional loss of certain attribute values due to the deletion of other data for other attributes. If we remove COMP10001 from the above table, the details of room EDS 6 are also deleted.

An **insertion** anomaly is the inability to add certain attributes to a database due to absence of other attributes. For example, a new room "NewAlice109" has been built but has not yet been timetabled for any course or members of staff.

If the relation is in denormalized form, it can lead to the above-mentioned anomalies. Normalization helps us remove such anomalies and get rid of redundant data by iteratively improving relations. This requires understanding of the functional dependencies of the attributes of a given relation and resolving transitive and partial dependencies to achieve normal forms.

Functional dependency

For a particular relation R, a functional dependency occurs when a subset of R's attributes $\{A_1, A_2, ..., A_n\}$ determine attributes $\{B_1, B_2, \ldots, B_n\}$. If several tuples agree on the values of $A_1, A_2, ..., A_n$, they must agree on the values of $B_1, B_2, \ldots, B_n$ – that is, if two records have the same $A_1, A_2, ..., A_n$ then they have the same $B_1, B_2, \ldots, B_n$. Therefore $A_1, A_2, ..., A_n$ "functionally determine" $B_1, B_2, \ldots, B_n$. This is written as:

$\mathbf{A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n}$

A relation R satisfies a functional dependency (FD) if and only if the FD is true for every instance of R.

Determinants

The attributes that determine the value of other attributes are called determinants. For example, consider the relation below:

Person (<u>ssn</u>, name, birthdate, address, age)

birthdate → age

ssn → name, birthdate, age, address

In this example, *birthdate* and *ssn* are determinants, as *birthdate* determines *age* and *ssn* determines the rest of the attributes.

Key and non-key attributes

A *key* is a set of attributes $\{A_1, A_2, \ldots, A_n\}$ for a relation R, such that $\{A_1, A_2, ..., A_n\}$ functionally determines all other attributes of R and no subset of $\{A_1, A_2, \ldots, A_n\}$ functionally determines all other attributes of R. The key must be minimal.

In the relation Person (ssn, name, birthdate, address, age), *ssn* is the minimal key of the Person relation, but {*ssn, name*} is not (it is a "super key").

Partial functional dependency

A partial functional dependency arises when one or more non-key attributes are functionally determined by a *subset* of the primary key. This is shown in Figure 1.

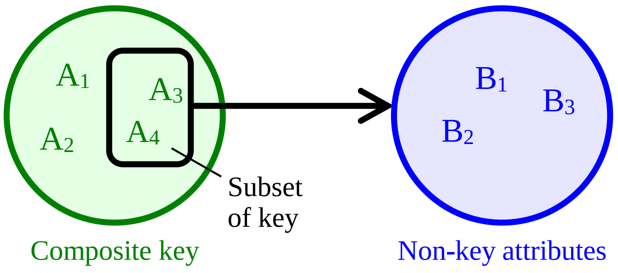


Figure 1: A partial functional dependency
(subset of composite key determining some non-key attributes)

Consider a relation R (A, B, C, D) with a composite primary key of (A, D), satisfying the functional dependencies A → B, D → C. For this relation, AD determines BC (AD → BC: AD can uniquely identify BC). However, to identify B, attribute A is enough and similarly D can identify C. Functional dependencies like A → B and D → C are called *partial functional dependencies*. For example, in the relation Order (Order#, Item#, Desc, Qty) from the lecture, Order# and Item# are the keys. Nonetheless, the item description, Desc, can be determined by Item# alone.

Transitive functional dependency

When a non-key attribute is determined by another non-key attribute (or by a subset of PK and non-key attributes), such a dependency is called a *transitive functional dependency*. This is shown in Figure 2.

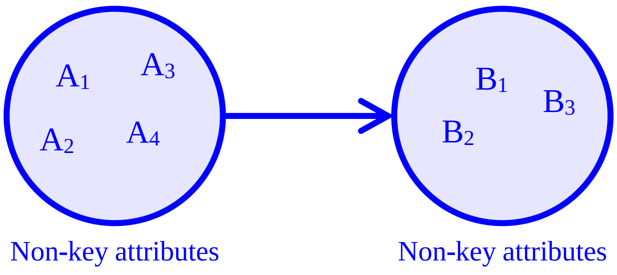


Figure 2: A transitive functional dependency
(non-key attributes determining other non-key attributes)

Armstrong’s Axioms

Given a relation and a set of functional dependencies (FDs), we can discover new functional dependencies using some rules generally known as Armstrong’s Axioms. Three important axioms required to determine FDs are Reflexivity (also known as “trivial FDs”), Augmentation and Transitivity.

Reflexivity: Suppose A = {A_1, A_2, ..., A_n} is a subset of attributes of R, and B = {B_1, B_2, …, B_n} is also a subset of attributes of R such that B is a subset of A. Reflexivity implies that

$$B \subseteq A \Longrightarrow A \rightarrow B$$

For example, in the case of Person (ssn, name, birthdate, address, age),

ssn, name → name

Augmentation: Consider another subset of attributes C = {C_1, C_2, …, C_n}. Augmentation implies that

$$A \rightarrow B \Longrightarrow AC \rightarrow BC$$

In the case of Person (ssn, name, birthdate, address, age), we can take the above FD and write

ssn, name, age → name, age

Transitivity:

$$A \rightarrow B \text{ and } B \rightarrow C \Longrightarrow A \rightarrow C$$

For our relation Person, we can say ssn → birthdate, birthdate → age ⟹ ssn → age.

Normalization Exercises

1.1 Consider the relation Diagnosis with the schema Diagnosis (DoctorID, DocName, PatientID, DiagnosisClass) and the following functional dependencies:

DoctorID → DocName
DoctorID, PatientID → DiagnosisClass

Consider the following instance of Diagnosis:

DoctorID	**DocName**	**PatientID**	**DiagnosisClass**
D001	Alicia	P888	Flu
D002	John	P999	Lactose intolerance
D003	Jennifer	P000	Flu
D002	John	P111	Fever

Identify different anomalies that can arise from this schema using the above instance.

1.2 Consider the following relation StaffPropertyInspection:

StaffPropertyInspection (propertyNo, pAddress, iDate, iTime, comments, staffNo, sName)

The FDs stated below hold for this relation:

propertyNo, iDate → iTime, comments, staffNo, sName
propertyNo → pAddress
staffNo → sName

From these FDs, it is safe to assume that propertyNo and iDate can serve as a primary key. Your task is to normalize this relation to 3NF. Remember in order to achieve 3NF, you first need to achieve 1NF and 2NF.

1.3 The following Report table is used by a publishing house to keep track of the editing and design of books by a number of authors:

report_no	**editor**	**dept_no**	**dept_name**	**dept_addr**	**author_id**	**auth_name**	**auth_addr**
4216	woolf	15	design	argus1	53	mantel	cs-tor
4216	woolf	15	design	argus1	44	bolton	mathrev
4216	woolf	15	design	argus1	71	koenig	mathrev
5789	koenig	27	analysis	argus2	26	fry	folkstone
5789	koenig	27	analysis	argus2	38	umar	prise
5789	koenig	27	analysis	argus2	71	koenig	mathrev

By looking at the data, we see that functional dependencies in the Report table are the following:

report_no → editor, dept_no

dept_no → dept_name, dept_addr

author_id → auth_name, author_addr

The candidate key for this relation is (report_no, author_id) since we need these two attributes to uniquely identify each record. Thus we have:

Report (<u>report_no</u>, editor, dept_no, dept_name, dept_addr, <u>author_id</u>, auth_name, auth_addr)

1.4 Is the Report table in 2NF? If not, put the table in 2NF.

1.5 Consider the following relation:

Class (<u>courseNumber</u>, roomNumber, instructorName, <u>studentNumber</u>, workshopNumber, grade, tutor)

The following functional dependencies hold for this relation:

workshopNumber → tutor

studentNumber, courseNumber → grade, workshopNumber

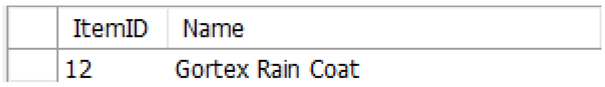
courseNumber → roomNumber, instructorName

Normalize this relation into 3NF.

Section 2 SQL

Connect to your MySQL database on the engineering server

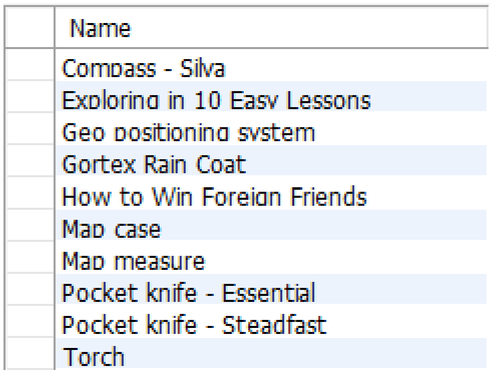
2.1. Type the query to list the green items of type C



ItemID	Name
12	Gortex Rain Coat

```
SELECT ItemID, Name
FROM ITEM
WHERE Type = 'C'
And Colour = 'Green';
```

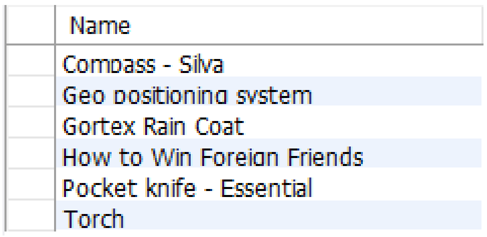
2.2 Type the query to find the items delivered by at least two suppliers



Name
Compass - Silva
Exploring in 10 Easy Lessons
Geo positioning system
Gortex Rain Coat
How to Win Foreign Friends
Map case
Map measure
Pocket knife - Essential
Pocket knife - Steadfast
Torch

```
SELECT item.Name
FROM item natural join deliveryitem natural join delivery
GROUP BY item.Name
HAVING COUNT(DISTINCT(SupplierID)) >= 2;
```

2.3 Type the query to find the items that have been sold by at least two departments

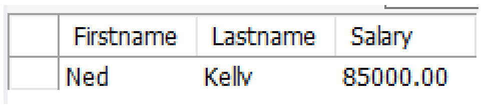


Name
Compass - Silva
Geo positioning system
Gortex Rain Coat
How to Win Foreign Friends
Pocket knife - Essential
Torch

```
SELECT item.Name
FROM item natural join saleitem natural join sale
GROUP BY item.Name
```

```
HAVING COUNT(DISTINCT(DepartmentID)) >= 2;
```

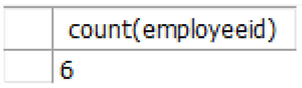
2.4 Find the name of the highest-paid employee in the Marketing department



Firstname	Lastname	Salary
Ned	Kelly	85000.00

```
SELECT employee.Firstname, employee.Lastname, employee.Salary
FROM employee natural join department
WHERE department.Name = 'Marketing'
AND employee.Salary =
    (SELECT max(salary)
    FROM employee natural join department
    WHERE department.Name = 'Marketing')
```

2.5 Type the query to find the number of employees with a salary equal to or less than $45,000

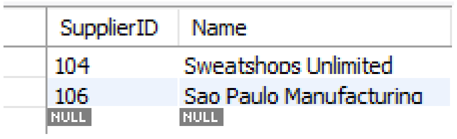


count(employeeid)
6

```
SELECT count(employeeid)
FROM employee
WHERE SALARY <= 45000;
```

2.6 Find the supplier id and supplier names that do not deliver compasses

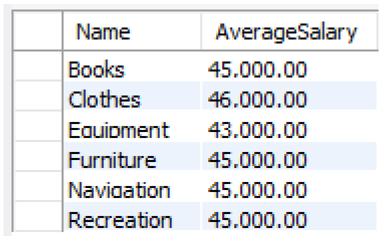
```
SELECT SupplierID, Supplier.Name
FROM Supplier
WHERE SupplierID NOT IN
     (SELECT SupplierID
      FROM Delivery NATURAL JOIN DeliveryItem NATURAL JOIN
ITEM
      WHERE Item.Name Like 'Compass%');
```



SupplierID	Name
104	Sweatshops Unlimited
106	Sao Paulo Manufacturing
NULL	NULL

2.7 Find, for each department that has sold items of type E. List the department name and the average salary of the employees

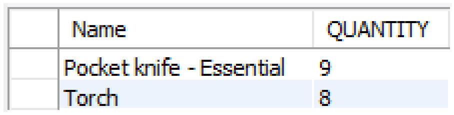
```
SELECT Department.Name, FORMAT(AVG(Employee.Salary),2) AS
AverageSalary
FROM Employee INNER JOIN Department INNER JOIN Sale
INNER JOIN SaleItem INNER JOIN Item
ON Employee.DepartmentID = Department.DepartmentID
AND Department.DepartmentID = Sale.DepartmentID
AND Sale.SaleID = SaleItem.SaleID
AND SaleItem.ItemID = Item.ItemID
WHERE Item.Type = 'E'
GROUP BY Department.Name;
```



Name	AverageSalary
Books	45.000.00
Clothes	46.000.00
Equipment	43.000.00
Furniture	45.000.00
Navigation	45.000.00
Recreation	45.000.00

2.8 Find the total number of items (list the item and sale quantity) of type E sold by the departments on the second floor

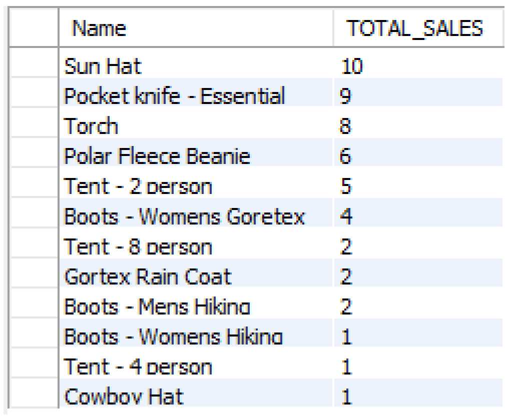
```
SELECT ITEM.Name, SUM(SaleItem.Quantity) AS QUANTITY
FROM Item INNER JOIN SaleItem INNER JOIN Sale INNER JOIN
Department
ON Item.ItemID = SaleItem.ItemID
AND Sale.SaleID = SaleItem.SaleID
AND Department.DepartmentID = Sale.DepartmentID
WHERE Item.Type = 'E'
AND Department.Floor = 2
GROUP BY ITEM.ITEMID;
```



Name	QUANTITY
Pocket knife - Essential	9
Torch	8

2.9 Type the query to find the total quantity sold of each item by the departments on the second floor

The result set should look similar to this:

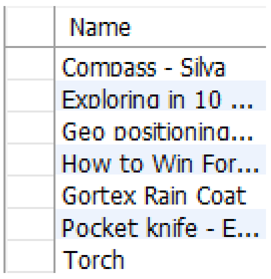


Name	TOTAL_SALES
Sun Hat	10
Pocket knife - Essential	9
Torch	8
Polar Fleece Beanie	6
Tent - 2 person	5
Boots - Womens Goretex	4
Tent - 8 person	2
Gortex Rain Coat	2
Boots - Mens Hiking	2
Boots - Womens Hiking	1
Tent - 4 person	1
Cowboy Hat	1

```
SELECT Item.Name, SUM(SaleItem.Quantity) as TOTAL_SALES
FROM Item INNER JOIN SaleItem INNER JOIN Sale INNER JOIN
Department
on Item.ItemiD = SaleItem.ItemID
AND SaleItem.SaleID = Sale.SaleID
AND Sale.DepartmentID = Department.DepartmentID
WHERE Department.Floor = 2
GROUP BY Item.Name
ORDER BY TOTAL_SALES DESC
```

2.10 List each item (ItemName) delivered to at least two departments by each supplier that delivers it

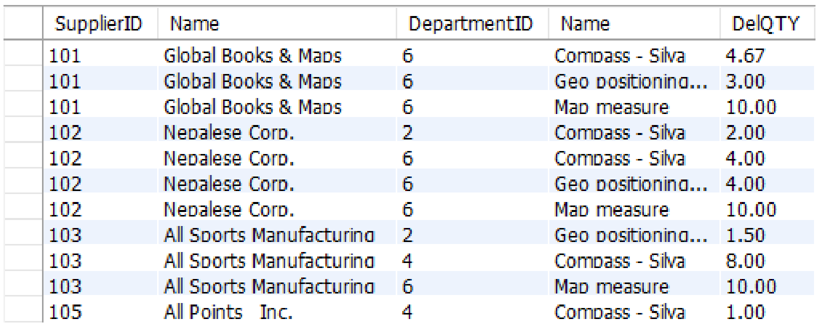
```
Select Distinct(Item.Name)
FROM Item natural join DeliveryItem
Where ItemID NOT IN (
 Select distinct(itemID)
 FRom DeliveryItem
 group by ItemID
 Having Count(distinct(departmentid)) < 2
 order by ItemID);
```



Name
Compass - Silva
Exploring in 10 ...
Geo positioning...
How to Win For...
Gortex Rain Coat
Pocket knife - E...
Torch

2.11 What is the average delivery quantity of items of type N delivered by each supplier to each department (given that the supplier delivers items of type N to the department)?

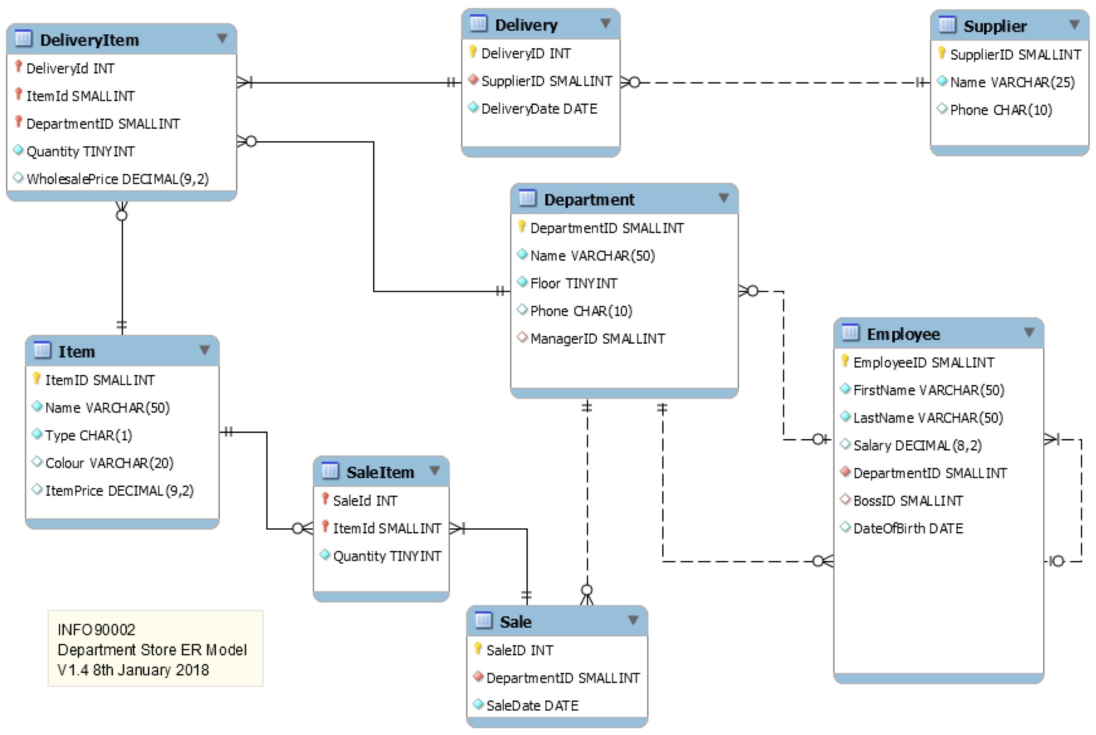
```
SELECT Delivery.SupplierID, Supplier.Name, DepartmentID,
Item.Name,
FORMAT(AVG(DeliveryItem.Quantity),2) AS DelQTY
FROM Delivery INNER JOIN Supplier INNER JOIN Item INNER JOIN
DeliveryItem
ON Delivery.SupplierID =  Supplier.SupplierID
AND DeliveryItem.ItemID = Item.ItemID
AND DeliveryItem.DeliveryID = Delivery.DeliveryID
WHERE  Item.Type = 'N'
GROUP BY Delivery.SupplierID,  Supplier.Name,
DepartmentID,  Item.Name;
```



SupplierID	Name	DepartmentID	Name	DelQTY
101	Global Books & Maps	6	Compass - Silva	4.67
101	Global Books & Maps	6	Geo positioning...	3.00
101	Global Books & Maps	6	Map measure	10.00
102	Nepalese Corp.	2	Compass - Silva	2.00
102	Nepalese Corp.	6	Compass - Silva	4.00
102	Nepalese Corp.	6	Geo positioning...	4.00
102	Nepalese Corp.	6	Map measure	10.00
103	All Sports Manufacturing	2	Geo positioning...	1.50
103	All Sports Manufacturing	4	Compass - Silva	8.00
103	All Sports Manufacturing	6	Map measure	10.00
105	All Points Inc.	4	Compass - Silva	1.00

End of Week 6 Tutorial

Appendix: The New Department Store Physical ER Model


DeliveryItem
DeliveryId INT
ItemId SMALLINT
DepartmentID SMALLINT
Quantity TINYINT
WholesalePrice DECIMAL(9,2)
Delivery
DeliveryID INT
SupplierID SMALLINT
DeliveryDate DATE
Supplier
SupplierID SMALLINT
Name VARCHAR(25)
Phone CHAR(10)
Department
DepartmentID SMALLINT
Name VARCHAR(50)
Floor TINYINT
Phone CHAR(10)
ManagerID SMALLINT
Employee
EmployeeID SMALLINT
FirstName VARCHAR(50)
LastName VARCHAR(50)
Salary DECIMAL(8,2)
DepartmentID SMALLINT
BossID SMALLINT
DateOfBirth DATE
Item
ItemID SMALLINT
Name VARCHAR(50)
Type CHAR(1)
Colour VARCHAR(20)
ItemPrice DECIMAL(9,2)
SaleItem
SaleId INT
ItemId SMALLINT
Quantity TINYINT
Sale
SaleID INT
DepartmentID SMALLINT
SaleDate DATE
INFO90002
Department Store ER Model
V1.4 8th January 2018