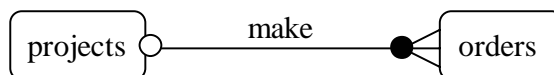


## Introduction to Database Systems

### TUTORIAL 5 – Transforming an E-R Model into a Relational Model

Transform the following E-R Models into a Relational Model [Note that the E-R Models presented are taken from the previous tutorial].

1. E-R Diagram



Entity-Sets

projects(projno, name, start\_date, end\_date)

orders(orderno, date, quantity)

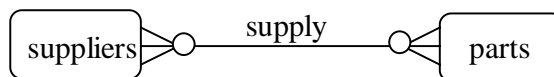
Relationships

make – projects make many orders [1:M][o:m]

Constraints/Assumptions

Project's end\_date must be after project's start\_date

2. E-R Diagram



Entity-Sets

suppliers(supplierno, name, tel\_no)

parts(partno, name, price)

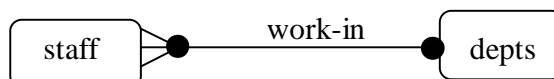
Relationships

supply – suppliers supply many parts [M:N][o:o]

Constraints/Assumptions

none

3. E-R Diagram



Entity-Sets

staff(staffid, name, address, phone\_no)

depts(deptid, name, location)

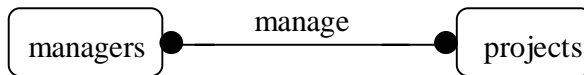
Relationships

work-in – many staff work in a departments [M:1][m:m]

Constraints/Assumptions

none

4. E-R Diagram



Entity-Sets

managers(manid, name, address, tel\_no)

projects(projid, name, start\_date, end\_date)

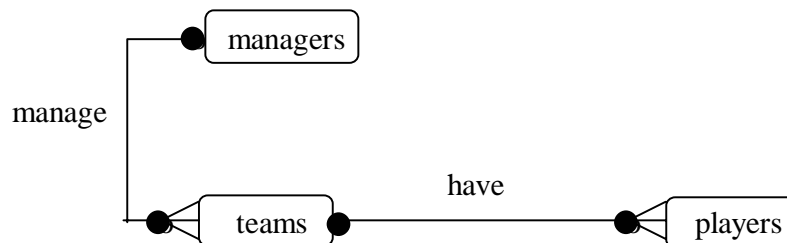
Relationships

manage – managers manage one project [1:1][m:m]

Constraints/Assumptions

Project's end\_date must be after project's start\_date

5. E-R Diagram



Entity-Sets

managers(manid, name, address, tel\_no)

teams(name, date\_founded, address)

players(playerid, name, address, tel\_no)

Relationships

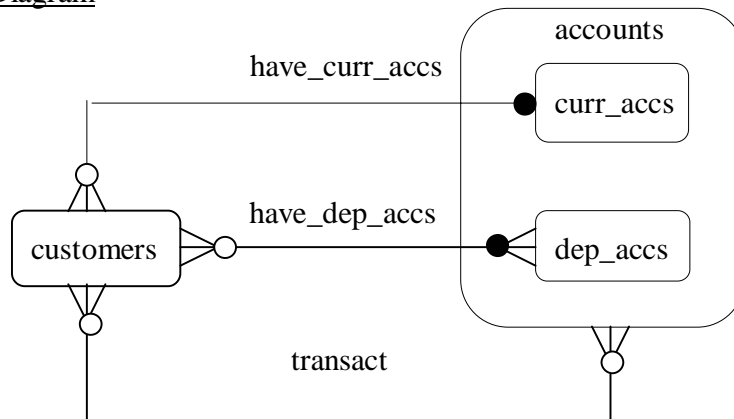
manage – managers manage many teams [1:M][m:m]

have – teams have many players [1:M][m:m]

Constraints/Assumptions

A player can only play in one team

6. E-R Diagram



Entity-Sets

customers(cust#, name, address, tel\_no)  
accounts(account#, name, current\_balance)  
curr\_accs(account#, credit\_limit)  
dep\_accs(account#, max\_withdrawals)

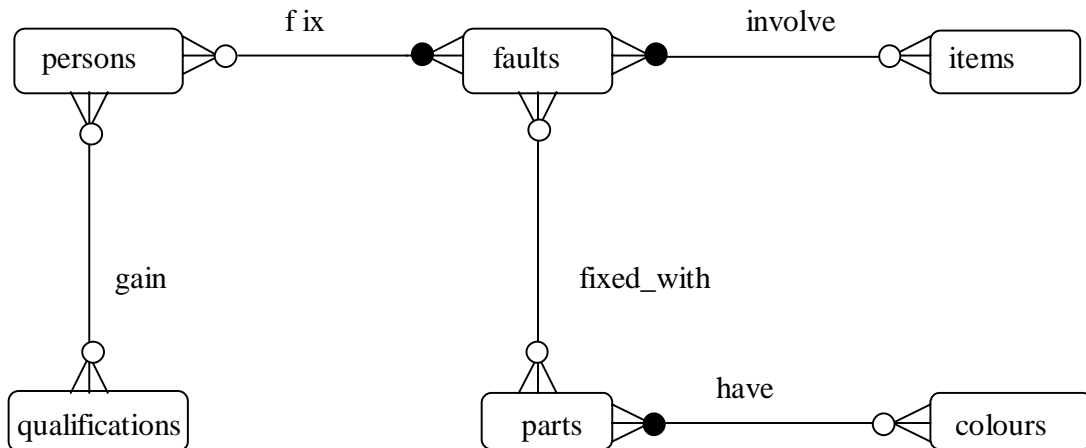
Relationships

have\_curr\_accs(cust# , account# , date\_opened)  
have\_dep\_accs(cust# , account# , date\_opened)  
transact(cust# , account# , amount, date)

Constraints/Assumptions

- (a) In the *curr\_accs* set we must always have  
 $current\_balance + credit\_limit > 0$
- (b) In the *dep\_accs* set we must always have  
 $current\_balance > 0$

7. E-R Diagram



Entity-Sets

items(item-no, location, type, description)  
 faults(fault-id, time-reported, time-fixed)  
 persons(person-id, surname, first-name)  
 qualifications(qual-code, q-description)  
 parts(part-id, weight, max-dimension)  
 colours(col-code, c-description)

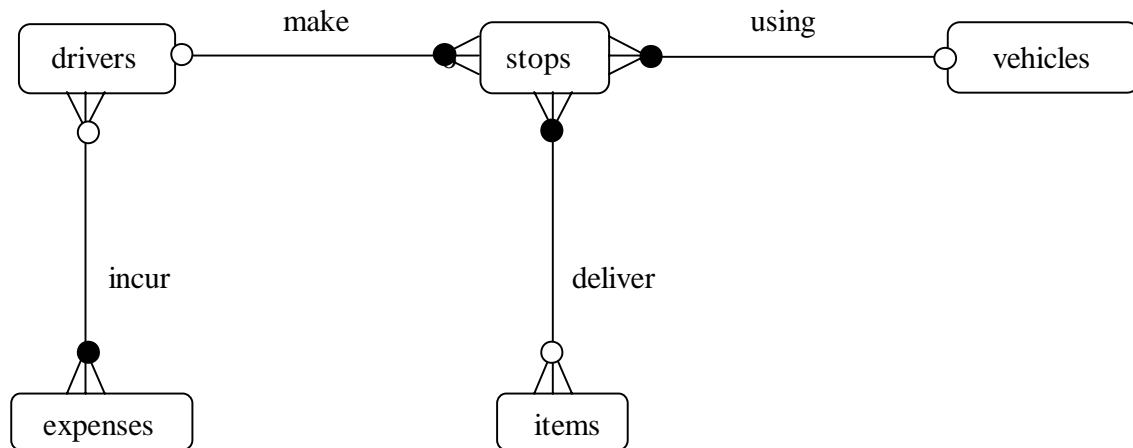
Relationships

gain(person-id, qual-code, date)  
 have – parts have many colours [M:N] [o:o]  
 involve – faults involve many items [M:N] [m:o]  
 fix(person-id, fault-id, time-spent)  
 fixed\_with(fault-id, part-id, qty-used)

Constraints/Assumptions

Each part's weight must be within a range of values

## 8. E-R Diagram



### Entity-Sets

drivers(driver-no, name, home-address, dob)

vehicles(reg-no, make, year)

items(item-no, color, weight, description)

stops(del-no, date, address)

expenses(type-no, description)

### Relationships

make(driver-no, del-no, date, time)

using(del-no, reg-no, date, time)

deliver(del-no, item-no, qty-left)

incur(driver-no, type-no, cost)

### Constraints/Assumptions

This database clearly needs tables to hold information on *drivers*, *vehicles*, and *items*. These tables will be linked by the stops made on route. At each stop the driver makes a delivery leaving quantities of various items. This delivery would presumably have a delivery note with a code number. This is not said in the description but is assumed in the solution.