Diaper

(http://sourceforge.org/projects/diaper)

Diaper, DIA Plugin for ER, is a tool that takes as input a ER model drawn in Dia 0.94 and converts into a SQL-92 based relational syntax.

The design of Diaper is quite elaborate, However, as of now it is limited by the features of Dia itself. Most notably, there is no elegant way to specify n-ary relationships in Dia. Also, there are no standard plugins to provide EER models.

How to use Diaper

Step 1. Using Dia to draw ER Diagrams

1. Start Dia. You should see two windows as shown.

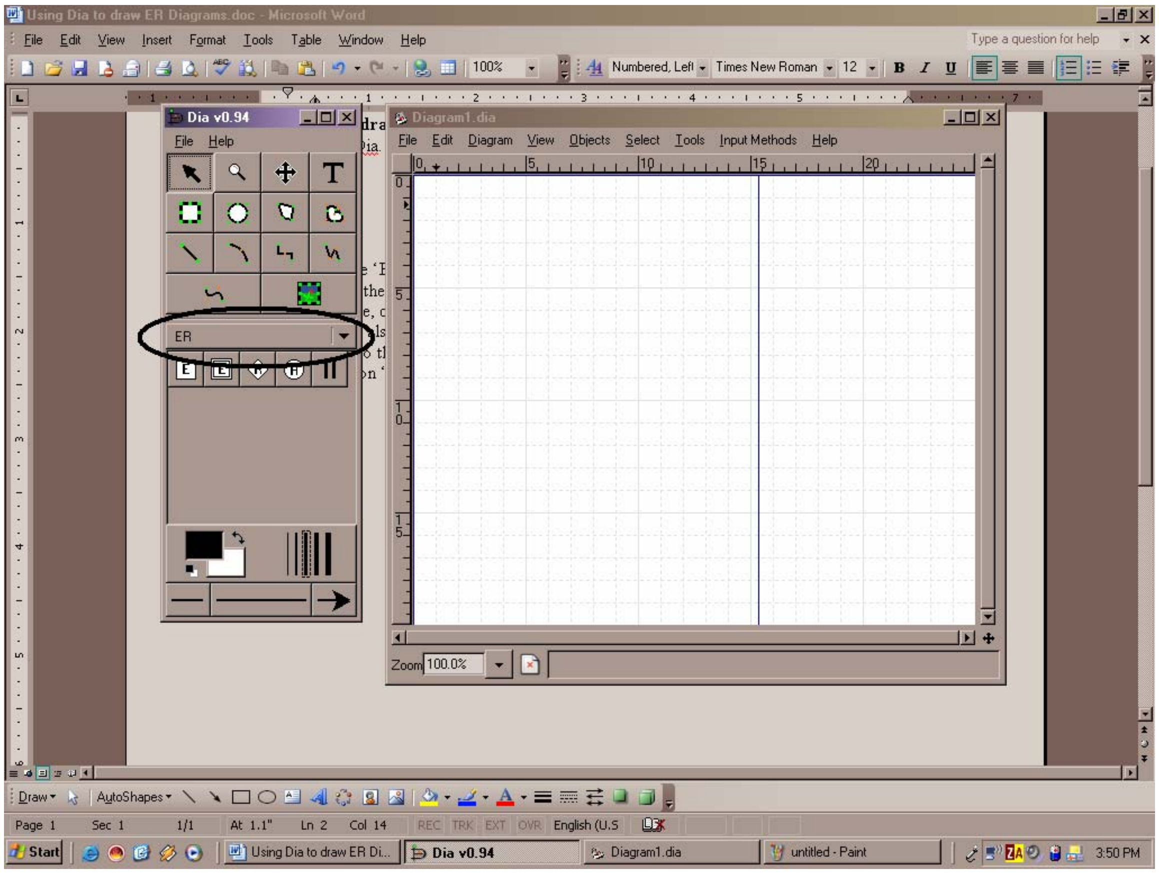

Using Dia to draw ER Diagrams.doc - Microsoft Word
Dia v0.94
File Help
ER
Diagram1.dia
File Edit Diagram View Objects Select Tools Input Methods Help
Zoom 100.0%
Start
Using Dia to draw ER Di...
Dia v0.94
Diagram1.dia
untitled - Paint
3:50 PM

Figure 1 – Dia windows and ER Drawing tool.

2. Choose ‘ER’ from the drop down menu in the Dia window. (Figure 1).
3. Select the required ER objects to draw an ER diagram in the Diagram window.
4. To save, click on ‘File’, select ‘Save As…’. Type in the required filename. You should also unselect a checkbox ‘Compress diagram file’. You must do this to ensure that the ER diagram is saved in XML format. (Figure 2 and Figure 3).
5. Click on ‘Ok’. This will save your ER Diagram file.

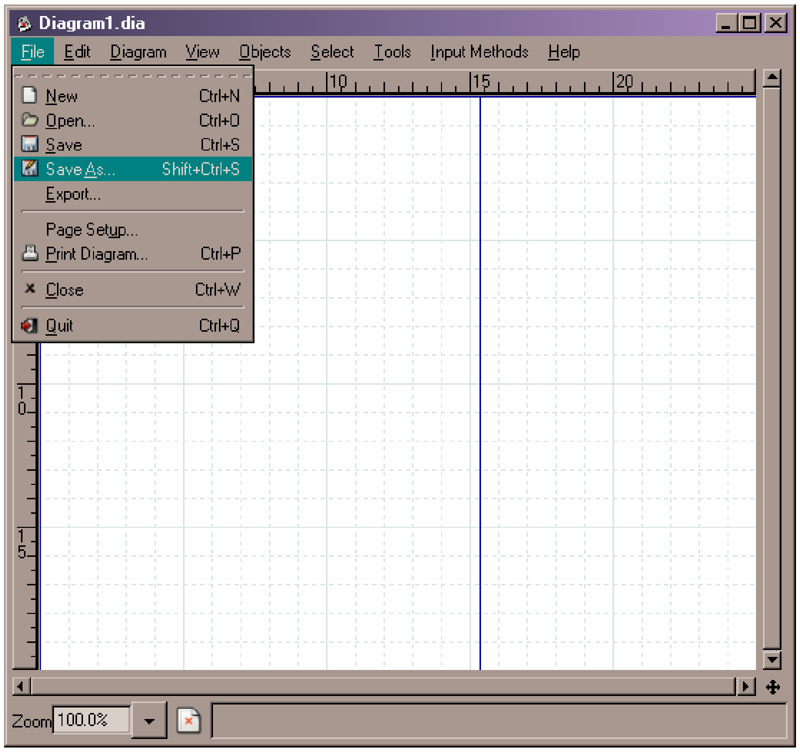
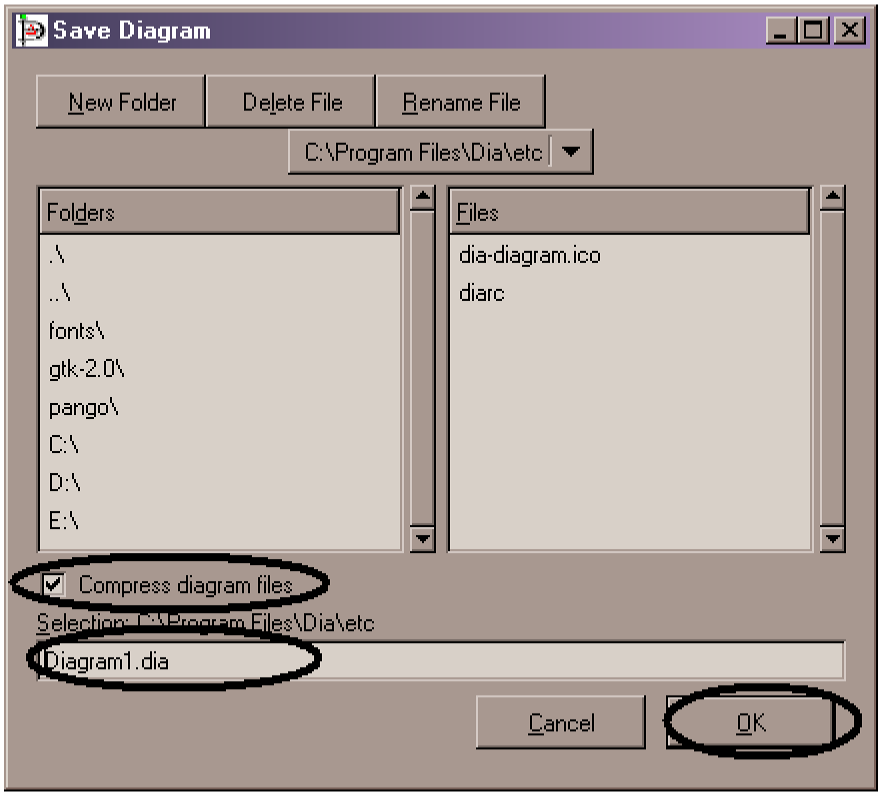


Figure 2 – Saving an ER Diagram in Dia.



[✓] Compress diagram files

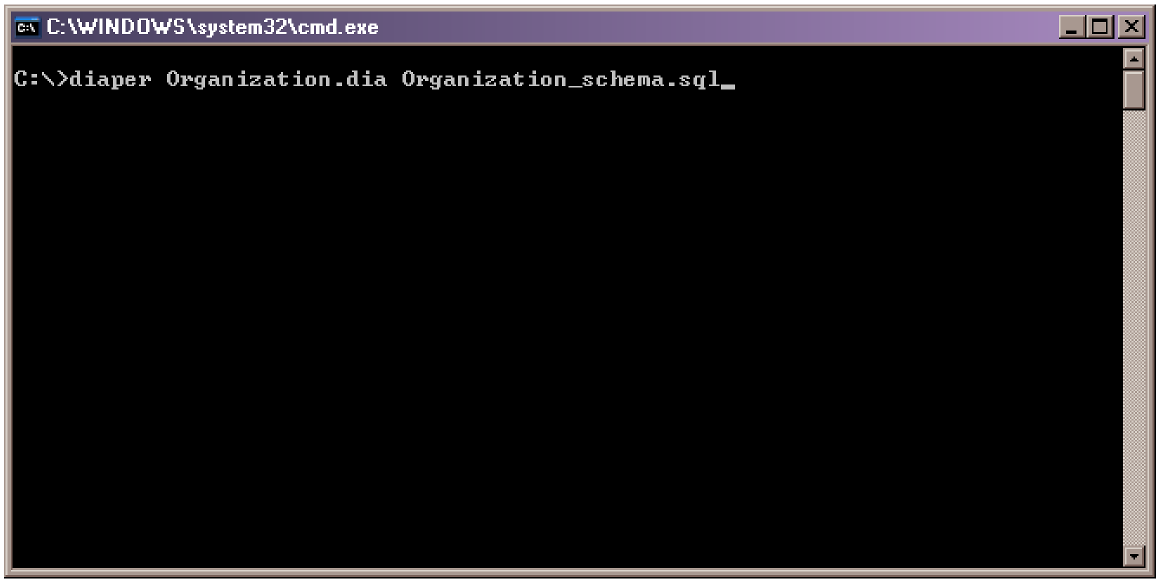
Selection: C:\Program Files\Dia\etc

Diagram1.dia

Figure 3 – Unchecking the Compress diagram files option

Step 2. Using Diaper to generate the schema for the ER Diagram

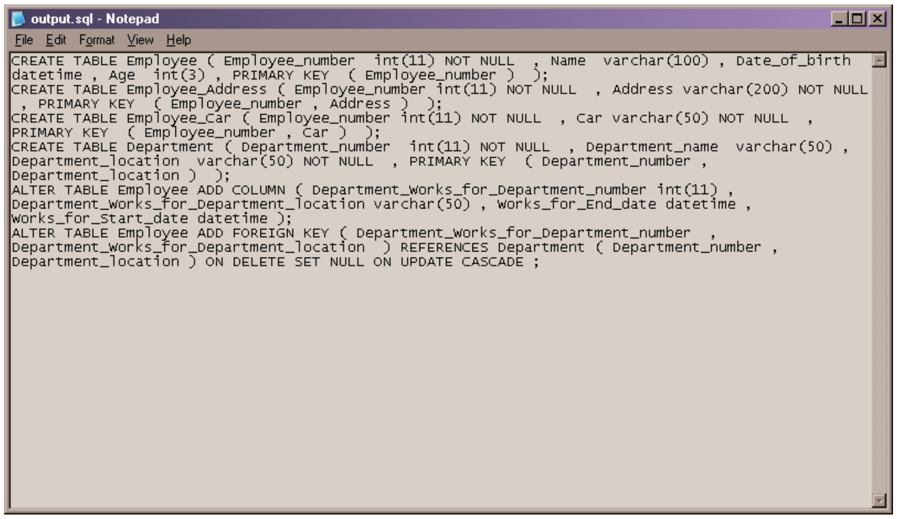
1. You need to open a command window.
2. Type Diaper followed by the input Dia file followed by the output file and press Enter. The input Dia file is the file that you have saved previously while drawing the ER diagram. The output filename is to be chosen by you. And example is shown below. (Figure 4).



```
C:\>diaper Organization.dia Organization_schema.sql_
```

Figure 4 – Generating the schema using Diaper

3. Open the output file to view the list of SQL statements required to generate the schema specified in the ER diagram. (Figure 5).



```
CREATE TABLE Employee ( Employee_number  int(11) NOT NULL  , Name  varchar(100) , Date_of_birth
datetime , Age  int(3) , PRIMARY KEY  ( Employee_number )  );
CREATE TABLE Employee_Address ( Employee_number int(11) NOT NULL  , Address varchar(200) NOT NULL
 , PRIMARY KEY  ( Employee_number , Address )  );
CREATE TABLE Employee_Car ( Employee_number int(11) NOT NULL  , Car varchar(50) NOT NULL  ,
PRIMARY KEY  ( Employee_number , Car )  );
CREATE TABLE Department ( Department_number  int(11) NOT NULL  , Department_name  varchar(50) ,
Department_location  varchar(50) NOT NULL  , PRIMARY KEY  ( Department_number ,
Department_location )  );
ALTER TABLE Employee ADD COLUMN ( Department_Works_for_Department_number int(11) ,
Department_Works_for_Department_location varchar(50) , Works_for_End_date datetime ,
Works_for_Start_date datetime );
ALTER TABLE Employee ADD FOREIGN KEY ( Department_Works_for_Department_number  ,
Department_Works_for_Department_location  ) REFERENCES Department ( Department_number ,
Department_location ) ON DELETE SET NULL ON UPDATE CASCADE ;
```

Figure 5 – Diaper generated SQL statement file

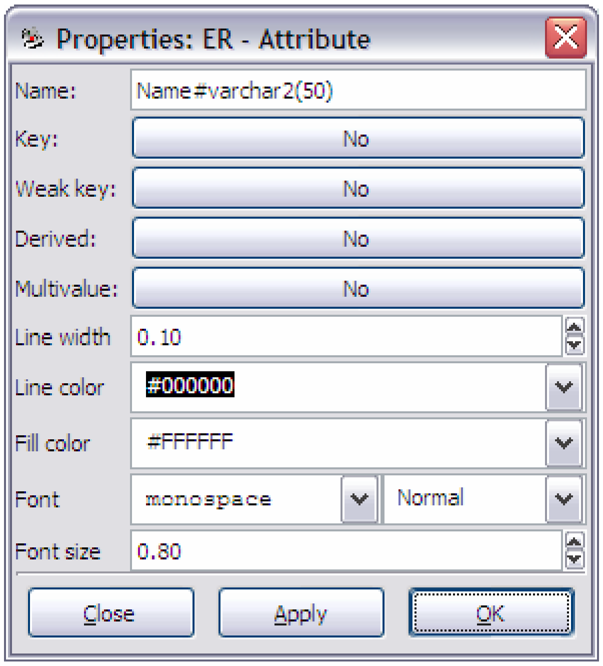
4. To generate the schema, copy the contents of the output Diaper file and paste in the database console.

Conventions

Because of some of the limitations of the Dia tool itself, some conventions have to be followed, most of these conventions are required to provide meta data, which is not present in ER, but is required when creating tables.

1) Specification of type for attributes: Attribute types are specified appending #type_Name [(size)] to the attribute name itself. So, if we have "name" attribute which must be of type varchar2(50), we will specify the attribute name as

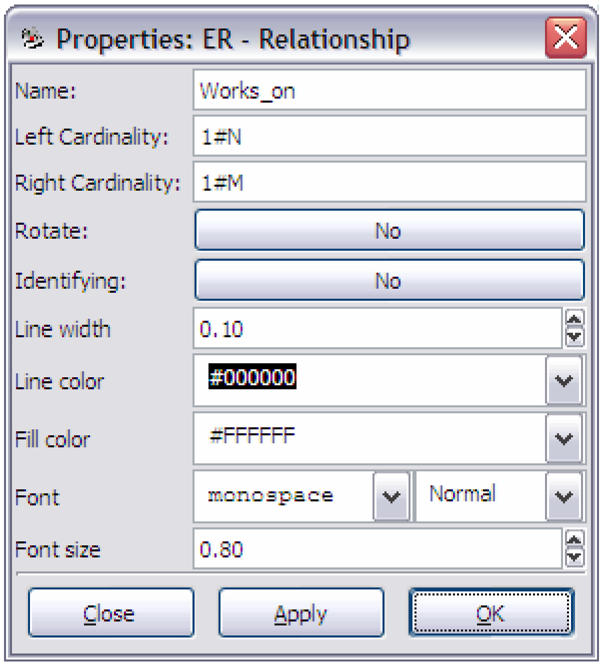
   Name#varchar2(50)



Name:	Name#varchar2(50)
Key:	No
Weak key:	No
Derived:	No
Multivalue:	No
Line width	0.10
Line color	#000000
Fill color	#FFFFFF
Font	monospace Normal
Font size	0.80

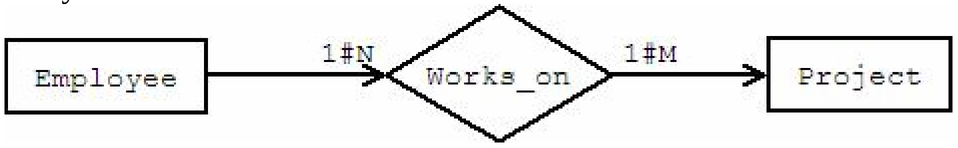
2) Specification of cardinality for relation: The cardinality of the relation is specified in the following format,
   a. 1:1 Relation can be specified by either
      i. Left Cardinality = 1 And Right Cardinality = 1
      ii. Left Cardinality = x#x And Right cardinality = x#x where x is positive integer.
   b. 1:N Relation can be specified by either
      i. Left Cardinality = 1 And Right Cardinality = N
      ii. Left Cardinality = x:x And Right Cardinality = N
      iii. Left Cardinality = x:x And Right Cardinality = x:y where x and y are positive integer, with y > x
   c. M:N relation can be specified by either.

i. Left Cardinality = M And Right Cardinality = N
ii. Left Cardinality = x:y And Right Cardinality = a:b, where a, b, x and y are positive integers with y > x and b > a.
iii. Left Cardinality = x:M And Right Cardinality = y:N, where x and y are positive integers.
iv. Left Cardinality = M And Right Cardinality = x:M where x is a positive integer.
v. Left Cardinality = x:M And Right Cardinality = N where x is a positive integer.


Properties: ER - Relationship

Name:	Works_on
Left Cardinality:	1#N
Right Cardinality:	1#M
Rotate:	No
Identifying:	No
Line width	0.10
Line color	#000000
Fill color	#FFFFFF
Font	monospace Normal
Font size	0.80

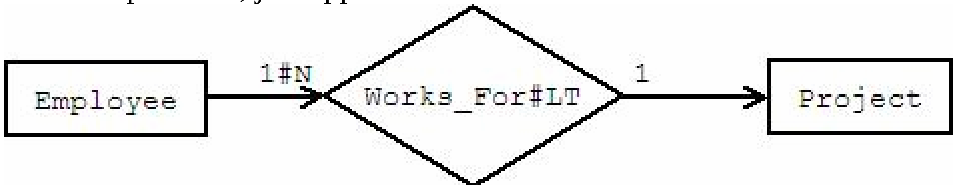
3) Participation of Relation: For output to be interpreted by DiaPER, the relation must be specified by having a line drawn into the relation from the left entity, and moving out towards the right
entity.



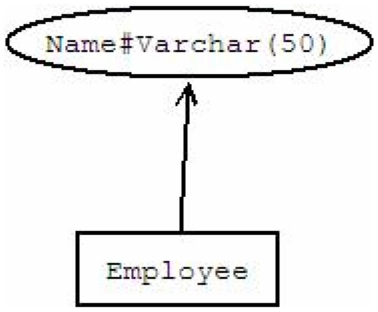
Here Employee is the left entity, while Project is the right entity. Having this convention is mandatory as there is no other way to interpret the left and right cardinality of the relation from the output that is generated by Dia.

4) Specification of Total Participation: In order to specify if a relation is participating totally either through left or right or both, the following convention is to followed.
a. To depict Left Participation, append #LT to the name of the relation.
b. To depict Right Participation, append #RT to the name of the relation.

c. To depict both, just append #LT#RT to the name of the relation.



5) Specification of attributes, for attributes the line must always be drawn from the entity towards the attribute.



Future Improvements

Future version of Diaper will provide the following

a) A normalization plugin, part of the work on it has already been done.
b) A better way to specify attribute types and total participation, this would require modifying the dia code itself.
c) A plugin for EER modeling.
d) Make Diaper more reliable and efficient.

Known issues

The following is the list of known issues

1) Diaper must be supplied with a correct **uncompressed** Dia 0.94 file, the behavior is undefined otherwise.
2) Diaper does not handle multi-valued attributes for weak entities.
3) Error handling can be improved.

Example schemas created using Dia and Diaper

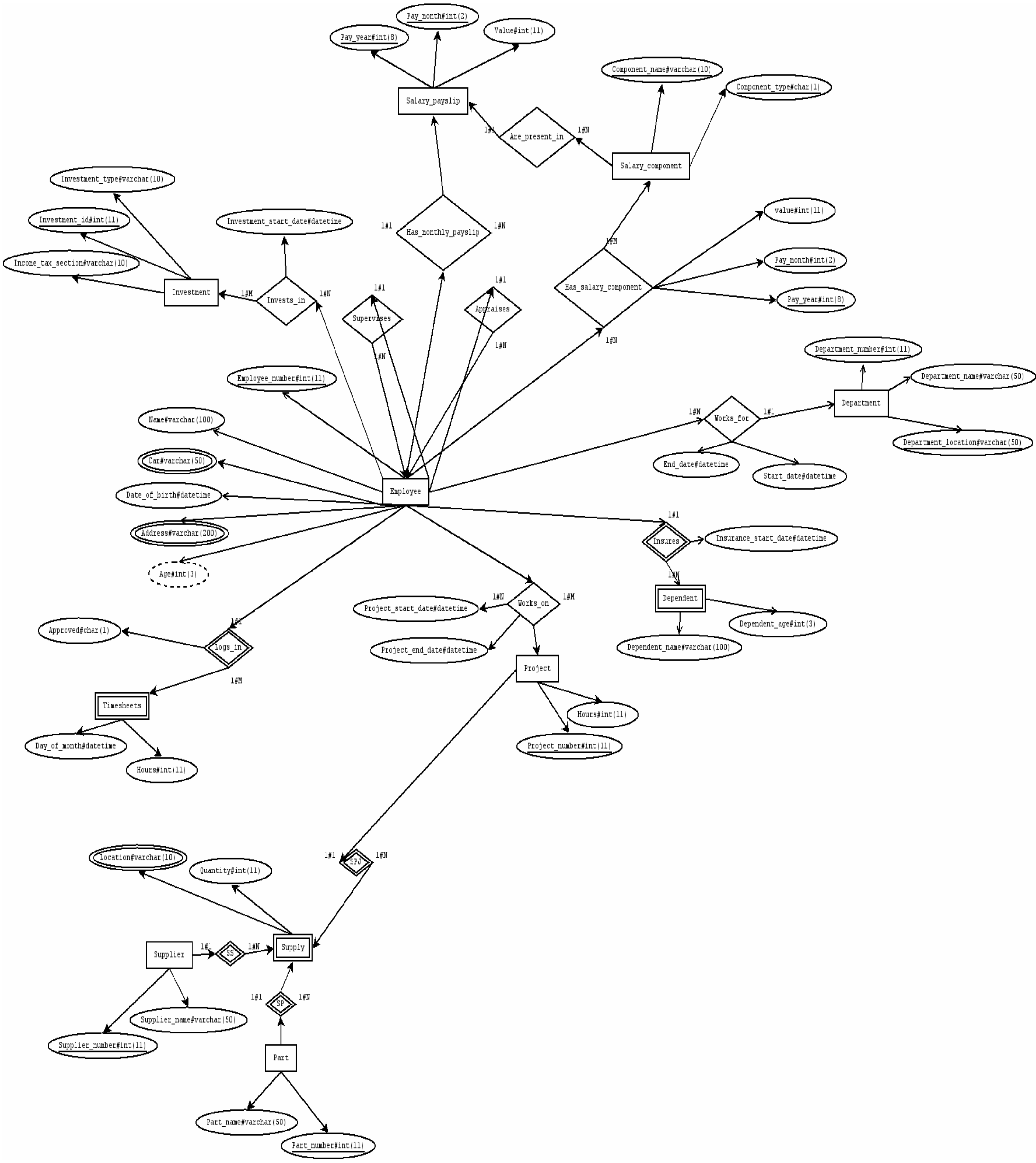
Example 1. This example schema (Figure 6) represents an organization with the following entities:

1. Employees
2. Departments
3. Dependents (Weak entity)
4. Projects
5. Salary components
6. Salary payslip
7. Dependents
8. Timesheets (Weak entity)
9. Investments
10. Supplier
11. Parts

The relations between the entities are:

1. Employee “Supervises” Employee (1:N)
2. Employee “Appraises” Employee (1:N)
3. Employee “Works for” Department (N:1)
4. Employee “Works on” Project (N:M)
5. Employee “Insures” Dependent (1:N), Dependent is a weak entity
6. Employee “Has Salary Component” Salary Component (N:M)
7. Employee “Has Monthly Payslip” Salary Payslip (1:N)
8. Salary Component “Are Present In” Salary Payslip (N:1)
9. Employee “Invests In” Investments (N:M)
10. Employee “Logs In” Timesheets (1:N), Timesheet is a weak entity
11. Supplier “Supplies” Part “for” Project (Ternary relationship expressed through binary relationships. Dia does not have relationship type to represent higher order relationships; they must be represented using weak entities and binary relationships.)

Figure 6 – Schema for an organization

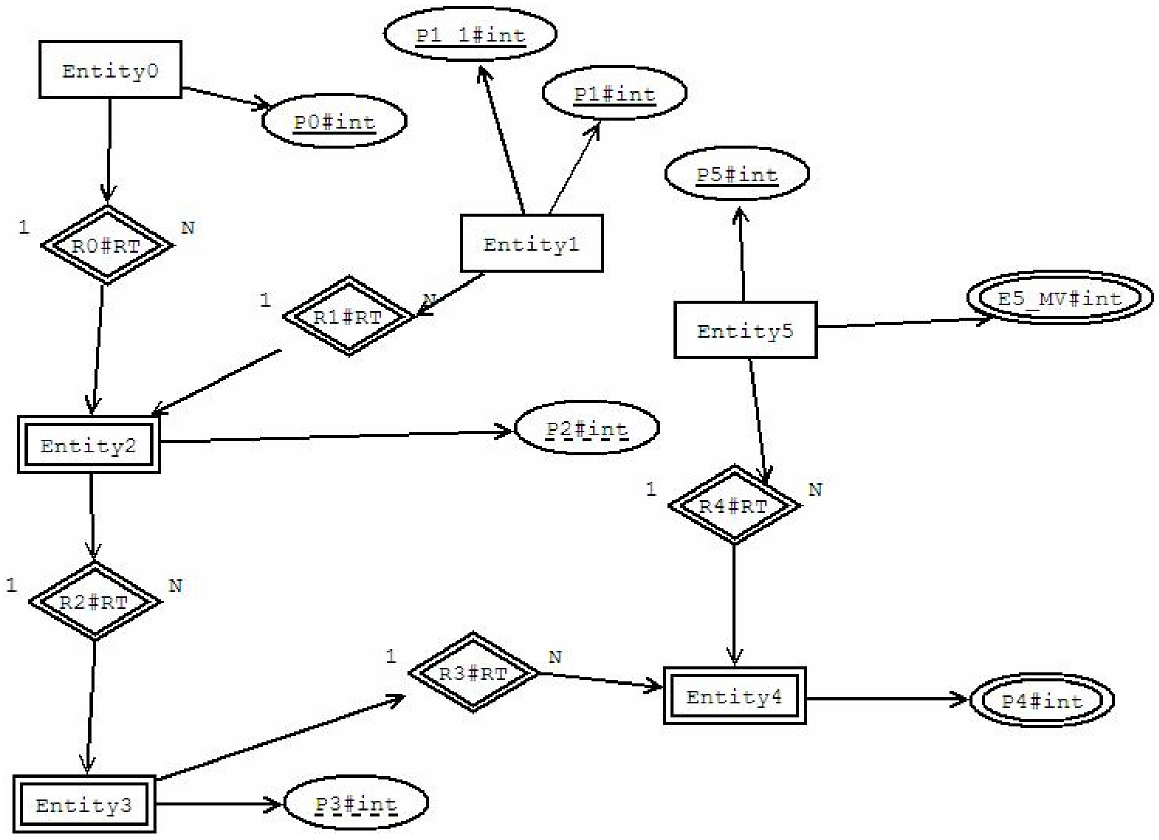


```
CREATE TABLE Employee ( Employee_number  int(11) NOT NULL  , Name
varchar(100) , Date_of_birth  datetime , Age  int(3) , PRIMARY KEY  (
Employee_number )  );
CREATE TABLE Employee_Address ( Employee_Employee_number int(11) NOT
NULL  , Employee_Address_Address varchar(200) NOT NULL  , PRIMARY KEY
( Employee_Employee_number , Employee_Address_Address )  );
ALTER TABLE Employee_Address ADD FOREIGN KEY  (
Employee_Employee_number ) REFERENCES Employee ( Employee_number )  ON
DELETE CASCADE ON UPDATE CASCADE ;
CREATE TABLE Employee_Car ( Employee_Employee_number int(11) NOT NULL
, Employee_Car_Car varchar(50) NOT NULL  , PRIMARY KEY  (
Employee_Employee_number , Employee_Car_Car )  );
ALTER TABLE Employee_Car ADD FOREIGN KEY  ( Employee_Employee_number )
REFERENCES Employee ( Employee_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
CREATE TABLE Department ( Department_number  int(11) NOT NULL  ,
Department_name  varchar(50) , Department_location  varchar(50) NOT
NULL  , PRIMARY KEY  ( Department_number , Department_location )  );
CREATE TABLE Dependent ( Employee_Employee_number  int(11) NOT NULL  ,
Dependent_name  varchar(100) , Dependent_age  int(3) , PRIMARY KEY  (
Employee_Employee_number )  );
CREATE TABLE Salary_component ( Component_name  varchar(10) NOT NULL  ,
Component_type  char(1) NOT NULL  , PRIMARY KEY  ( Component_name ,
Component_type )  );
CREATE TABLE Investment ( Investment_id  int(11) NOT NULL  ,
Investment_type  varchar(10) , Income_tax_section  varchar(10) ,
PRIMARY KEY  ( Investment_id )  );
CREATE TABLE Project ( Project_number  int(11) NOT NULL  , Hours
int(11) , PRIMARY KEY  ( Project_number )  );
CREATE TABLE Timesheets ( Employee_Employee_number  int(11) NOT NULL  ,
Day_of_month  datetime , Hours  int(11) , PRIMARY KEY  (
Employee_Employee_number )  );
CREATE TABLE Salary_payslip ( Pay_month  int(2) NOT NULL  , Pay_year
int(8) NOT NULL  , Value  int(11) , PRIMARY KEY  ( Pay_month , Pay_year
)  );
CREATE TABLE Supplier ( Supplier_number  int(11) NOT NULL  ,
Supplier_name  varchar(50) , PRIMARY KEY  ( Supplier_number )  );
CREATE TABLE Supply ( Project_Project_number  int(11) NOT NULL  ,
Part_Part_number  int(11) NOT NULL  , Supplier_Supplier_number  int(11)
NOT NULL  , Quantity  int(11) , PRIMARY KEY  ( Supplier_Supplier_number
, Part_Part_number , Project_Project_number )  );
CREATE TABLE Part ( Part_number  int(11) NOT NULL  , Part_name
varchar(50) , PRIMARY KEY  ( Part_number )  );
ALTER TABLE Employee ADD COLUMN (Department_Department_number int(11) ,
Department_Department_location varchar(50) );
ALTER TABLE Employee ADD COLUMN (Works_for_End_date datetime ,
Works_for_Start_date datetime );
ALTER TABLE Employee ADD FOREIGN KEY  ( Department_Department_number ,
Department_Department_location ) REFERENCES Department (
Department_number , Department_location )  ON DELETE SET NULL ON UPDATE
CASCADE ;
ALTER TABLE Dependent ADD COLUMN (Insures_Insurance_start_date datetime
);
ALTER TABLE Dependent ADD FOREIGN KEY  ( Employee_Employee_number )
REFERENCES Employee ( Employee_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
```

```
CREATE TABLE Has_salary_component ( Employee_Employee_number int(11)
NOT NULL  , Salary_component_Component_name varchar(10) NOT NULL  ,
Salary_component_Component_type char(1) NOT NULL  ,
Has_salary_component_Pay_month int(2) NOT NULL  ,
Has_salary_component_Pay_year int(8) NOT NULL  ,
Has_salary_component_value int(11) , PRIMARY KEY  (
Employee_Employee_number , Salary_component_Component_name ,
Salary_component_Component_type , Has_salary_component_Pay_month ,
Has_salary_component_Pay_year )  );
ALTER TABLE Has_salary_component ADD FOREIGN KEY  (
Employee_Employee_number ) REFERENCES Employee ( Employee_number )  ON
DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Has_salary_component ADD FOREIGN KEY  (
Salary_component_Component_name , Salary_component_Component_type )
REFERENCES Salary_component ( Component_name , Component_type )  ON
DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Employee ADD COLUMN (Supervises_Employee_Employee_number
int(11) );
ALTER TABLE Employee ADD FOREIGN KEY  (
Supervises_Employee_Employee_number ) REFERENCES Employee (
Employee_number )  ON DELETE SET NULL ON UPDATE CASCADE ;
CREATE TABLE Invests_in ( Employee_Employee_number int(11) NOT NULL  ,
Investment_Investment_id int(11) NOT NULL  ,
Invests_in_Investment_start_date datetime , PRIMARY KEY  (
Employee_Employee_number , Investment_Investment_id )  );
ALTER TABLE Invests_in ADD FOREIGN KEY  ( Employee_Employee_number )
REFERENCES Employee ( Employee_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Invests_in ADD FOREIGN KEY  ( Investment_Investment_id )
REFERENCES Investment ( Investment_id )  ON DELETE CASCADE ON UPDATE
CASCADE ;
CREATE TABLE Works_on ( Employee_Employee_number int(11) NOT NULL  ,
Project_Project_number int(11) NOT NULL  , Works_on_Project_start_date
datetime , Works_on_Project_end_date datetime , PRIMARY KEY  (
Employee_Employee_number , Project_Project_number )  );
ALTER TABLE Works_on ADD FOREIGN KEY  ( Employee_Employee_number )
REFERENCES Employee ( Employee_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Works_on ADD FOREIGN KEY  ( Project_Project_number )
REFERENCES Project ( Project_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Timesheets ADD COLUMN (Logs_in_Approved char(1) );
ALTER TABLE Timesheets ADD FOREIGN KEY  ( Employee_Employee_number )
REFERENCES Employee ( Employee_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Employee ADD COLUMN (Appraises_Employee_Employee_number
int(11) );
ALTER TABLE Employee ADD FOREIGN KEY  (
Appraises_Employee_Employee_number ) REFERENCES Employee (
Employee_number )  ON DELETE SET NULL ON UPDATE CASCADE ;
ALTER TABLE Salary_payslip ADD COLUMN (Employee_Employee_number int(11)
);
ALTER TABLE Salary_payslip ADD FOREIGN KEY  ( Employee_Employee_number
) REFERENCES Employee ( Employee_number )  ON DELETE SET NULL ON UPDATE
CASCADE ;
ALTER TABLE Salary_payslip ADD COLUMN (Salary_component_Component_name
varchar(10) , Salary_component_Component_type char(1) );
```

```
ALTER TABLE Salary_payslip ADD FOREIGN KEY  (
Salary_component_Component_name , Salary_component_Component_type )
REFERENCES Salary_component ( Component_name , Component_type )  ON
DELETE SET NULL ON UPDATE CASCADE ;
ALTER TABLE Supply ADD FOREIGN KEY  ( Part_Part_number ) REFERENCES
Part ( Part_number )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Supply ADD FOREIGN KEY  ( Project_Project_number )
REFERENCES Project ( Project_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Supply ADD FOREIGN KEY  ( Supplier_Supplier_number )
REFERENCES Supplier ( Supplier_number )  ON DELETE CASCADE ON UPDATE
CASCADE ;
```

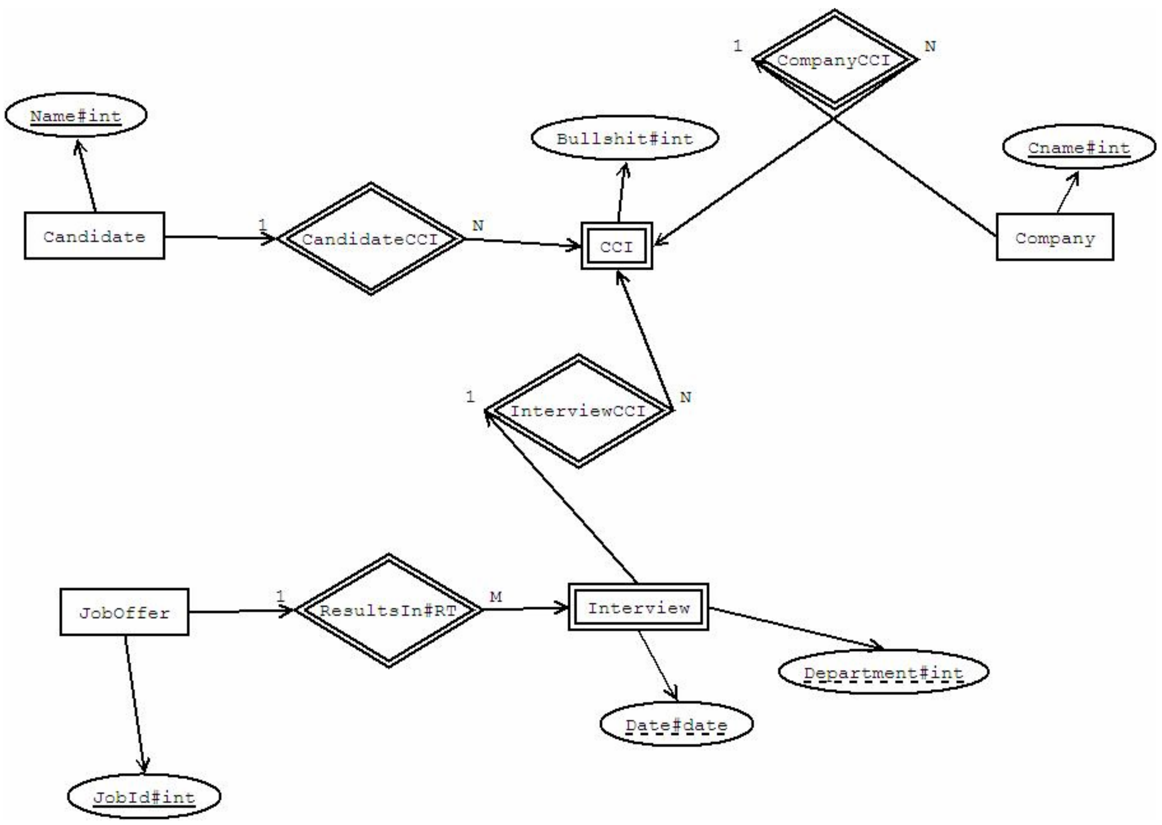
ER Diagram that describes weak entities with multi level cascading. (Figure 7)



Output Generated

```
CREATE TABLE Entity2 ( P2  int , PRIMARY KEY  ( P2 )  );
CREATE TABLE Entity0 ( P0  int NOT NULL  , PRIMARY KEY  ( P0 )  );
CREATE TABLE Entity1 ( P1  int NOT NULL  , PRIMARY KEY  ( P1 )  );
CREATE TABLE Entity3 ( P3  int , PRIMARY KEY  ( P3 )  );
CREATE TABLE Entity4 ( P4  int , PRIMARY KEY  ( P4 )  );
CREATE TABLE Entity5 ( P5  int NOT NULL  , PRIMARY KEY  ( P5 )  );
CREATE TABLE Entity5_E5_Multi_Val ( Entity5_E5_Multi_Val_P5 int NOT
NULL  , Entity5_E5_Multi_Val_E5_Multi_Val int NOT NULL  , PRIMARY KEY
( Entity5_E5_Multi_Val_P5 , Entity5_E5_Multi_Val_E5_Multi_Val )  );
ALTER TABLE Entity5_E5_Multi_Val ADD FOREIGN KEY  (
Entity5_E5_Multi_Val_P5 ) REFERENCES Entity5 ( P5 ) ;
ALTER TABLE Entity2 ADD COLUMN ( Entity0_R0_P0 int );
ALTER TABLE Entity2 ADD FOREIGN KEY  ( Entity0_R0_P0 ) REFERENCES
Entity0 ( P0 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity2 ADD COLUMN ( Entity1_R1_P1 int );
ALTER TABLE Entity2 ADD FOREIGN KEY  ( Entity1_R1_P1 ) REFERENCES
Entity1 ( P1 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity3 ADD COLUMN ( Entity2_R2_P2 int , Entity0_R2_P0 int
, Entity1_R2_P1 int );
ALTER TABLE Entity3 ADD FOREIGN KEY  ( Entity2_R2_P2 ) REFERENCES
Entity2 ( P2 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity3 ADD FOREIGN KEY  ( Entity0_R2_P0 ) REFERENCES
Entity0 ( P0 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity3 ADD FOREIGN KEY  ( Entity1_R2_P1 ) REFERENCES
Entity1 ( P1 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity4 ADD COLUMN ( Entity3_R3_P3 int , Entity2_R3_P2 int
, Entity0_R3_P0 int , Entity1_R3_P1 int );
ALTER TABLE Entity4 ADD FOREIGN KEY  ( Entity3_R3_P3 ) REFERENCES
Entity3 ( P3 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity4 ADD FOREIGN KEY  ( Entity2_R3_P2 ) REFERENCES
Entity2 ( P2 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity4 ADD FOREIGN KEY  ( Entity0_R3_P0 ) REFERENCES
Entity0 ( P0 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity4 ADD FOREIGN KEY  ( Entity1_R3_P1 ) REFERENCES
Entity1 ( P1 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
ALTER TABLE Entity4 ADD COLUMN ( Entity5_R4_P5 int );
ALTER TABLE Entity4 ADD FOREIGN KEY  ( Entity5_R4_P5 ) REFERENCES
Entity5 ( P5 )  ON DELETE CASCADE ON UPDATE CASCADE  ;
```

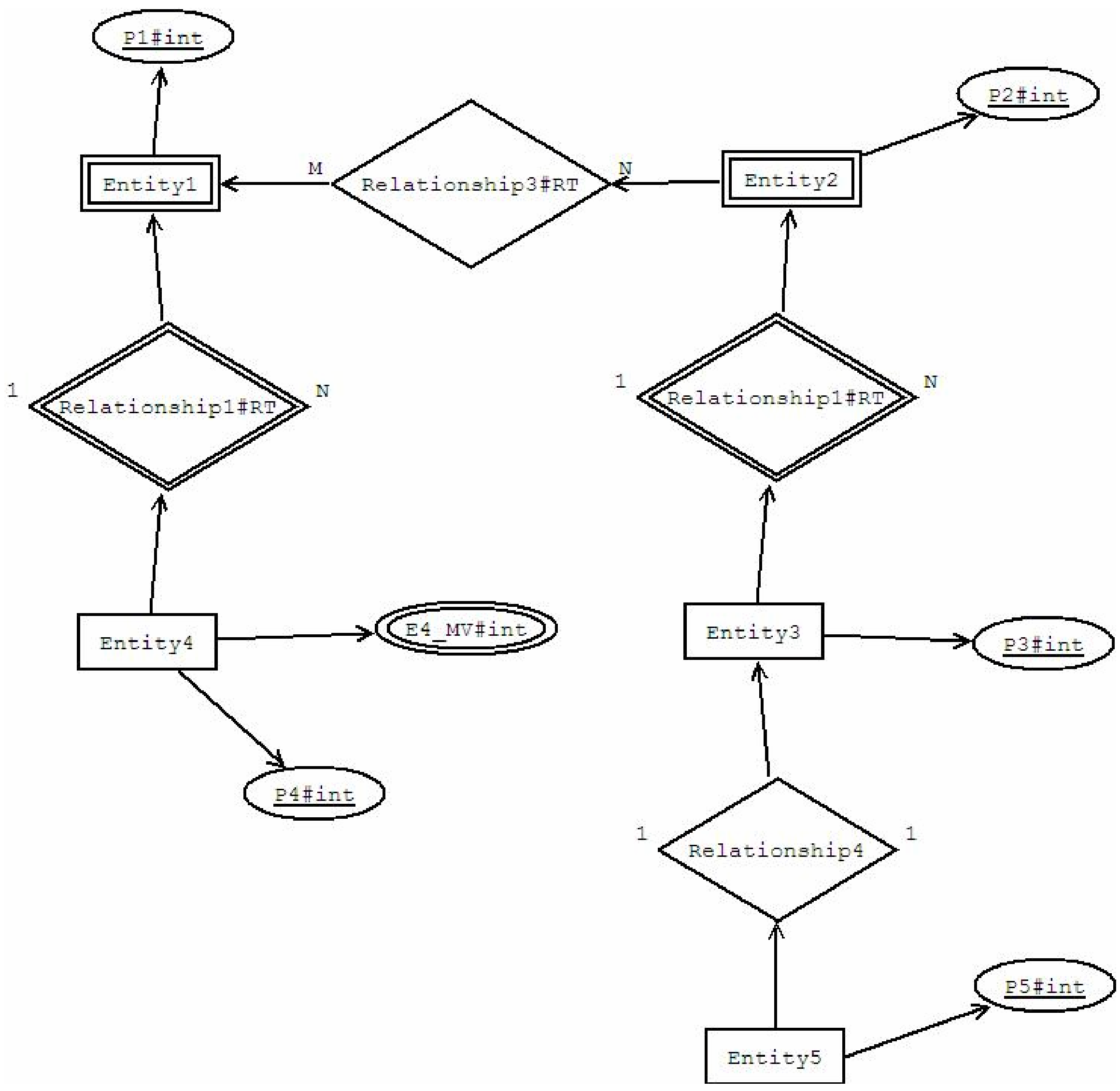
ER diagram describing interview process (Figure 8)



Output Generated:

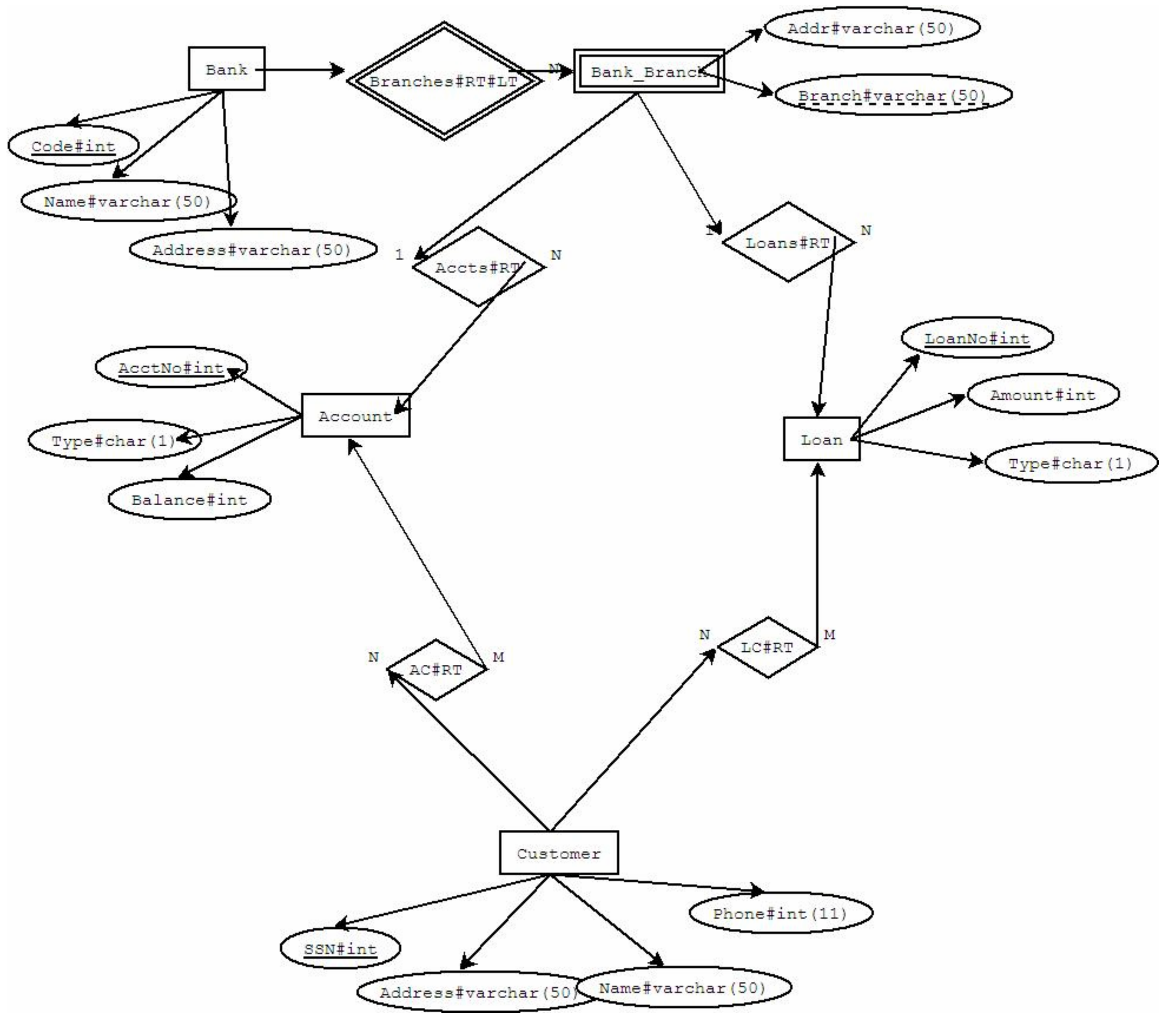
```
CREATE TABLE Candidate ( Name  int NOT NULL  , PRIMARY KEY  ( Name )
);
CREATE TABLE CCI ( Interview_Date  date , Candidate_Name  int NOT NULL
, JobOffer_JobId  int NOT NULL  , Interview_Department  int ,
Company_Cname  int NOT NULL  , Bullshit  int , PRIMARY KEY  (
Candidate_Name , Company_Cname , Interview_Date , Interview_Department
, JobOffer_JobId )  );
CREATE TABLE JobOffer ( JobId  int NOT NULL  , PRIMARY KEY  ( JobId )
);
CREATE TABLE Interview ( Date  date , JobOffer_JobId  int NOT NULL  ,
Department  int , PRIMARY KEY  ( Date , Department , JobOffer_JobId )
);
CREATE TABLE Company ( Cname  int NOT NULL  , PRIMARY KEY  ( Cname )
);
ALTER TABLE CCI ADD FOREIGN KEY  ( Candidate_Name ) REFERENCES
Candidate ( Name )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE CCI ADD FOREIGN KEY  ( Company_Cname ) REFERENCES Company (
Cname )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE CCI ADD FOREIGN KEY  ( Interview_Date ,
Interview_Department , JobOffer_JobId ) REFERENCES Interview ( Date ,
Department , JobOffer_JobId )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Interview ADD FOREIGN KEY  ( JobOffer_JobId ) REFERENCES
JobOffer ( JobId )  ON DELETE CASCADE ON UPDATE CASCADE ;
```

A complex scenario of weak entities (Figure 9)



```
CREATE TABLE Entity1 ( P1  int NOT NULL  , Entity4_P4  int NOT NULL  ,
PRIMARY KEY  ( P1 , Entity4_P4 )  );
CREATE TABLE Entity2 ( P2  int NOT NULL  , Entity3_P3  int NOT NULL  ,
PRIMARY KEY  ( P2 , Entity3_P3 )  );
CREATE TABLE Entity3 ( P3  int NOT NULL  , PRIMARY KEY  ( P3 )  );
CREATE TABLE Entity4 ( P4  int NOT NULL  , PRIMARY KEY  ( P4 )  );
CREATE TABLE Entity4_E4_MV ( Entity4_P4 int NOT NULL  ,
Entity4_E4_MV_E4_MV int NOT NULL  , PRIMARY KEY  ( Entity4_P4 ,
Entity4_E4_MV_E4_MV )  );
ALTER TABLE Entity4_E4_MV ADD FOREIGN KEY  ( Entity4_P4 ) REFERENCES
Entity4 ( P4 )  ON DELETE CASCADE ON UPDATE CASCADE ;
CREATE TABLE Entity5 ( P5  int NOT NULL  , PRIMARY KEY  ( P5 )  );
CREATE TABLE Relationship3 ( Entity2_P2 int NOT NULL  , Entity3_P3 int
NOT NULL  , Entity1_P1 int NOT NULL  , Entity4_P4 int NOT NULL  ,
PRIMARY KEY  ( Entity2_P2 , Entity3_P3 , Entity1_P1 , Entity4_P4 )  );
ALTER TABLE Relationship3 ADD FOREIGN KEY  ( Entity2_P2 , Entity3_P3 )
REFERENCES Entity2 ( P2 , Entity3_P3 )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Relationship3 ADD FOREIGN KEY  ( Entity1_P1 , Entity4_P4 )
REFERENCES Entity1 ( P1 , Entity4_P4 )  ON DELETE CASCADE ON UPDATE
CASCADE ;
ALTER TABLE Entity2 ADD FOREIGN KEY  ( Entity3_P3 ) REFERENCES Entity3
( P3 )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Entity1 ADD FOREIGN KEY  ( Entity4_P4 ) REFERENCES Entity4
( P4 )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Entity3 ADD COLUMN (Entity5_P5 int );
ALTER TABLE Entity3 ADD FOREIGN KEY  ( Entity5_P5 ) REFERENCES Entity5
( P5 )  ON DELETE SET NULL ON UPDATE CASCADE ;
```

ER diagram describing a small bank (Figure 10)



```
CREATE TABLE Bank ( Code  int NOT NULL  , Name  varchar(50) , Address
varchar(50) , PRIMARY KEY  ( Code )  );
CREATE TABLE Bank_Branch ( Bank_Code  int NOT NULL  , Addr  varchar(50)
, Branch  varchar(50) , PRIMARY KEY  ( Branch , Bank_Code )  );
CREATE TABLE Account ( AcctNo  int NOT NULL  , Type  char(1) , Balance
int , PRIMARY KEY  ( AcctNo )  );
CREATE TABLE Loan ( LoanNo  int NOT NULL  , Amount  int , Type  char(1)
, PRIMARY KEY  ( LoanNo )  );
CREATE TABLE Customer ( SSN  int NOT NULL  , Address  varchar(50) ,
Name  varchar(50) , Phone  int(11) , PRIMARY KEY  ( SSN )  );
ALTER TABLE Bank_Branch ADD FOREIGN KEY  ( Bank_Code ) REFERENCES Bank
( Code )  ON DELETE CASCADE ON UPDATE CASCADE ;
ALTER TABLE Account ADD COLUMN (Bank_Branch_Branch varchar(50) ,
Bank_Code int );
```

```
ALTER TABLE Account ADD FOREIGN KEY  ( Bank_Branch_Branch , Bank_Code )
REFERENCES Bank_Branch ( Branch , Bank_Code )  ON DELETE CASCADE ON
UPDATE CASCADE ;
ALTER TABLE Loan ADD COLUMN (Bank_Branch_Branch varchar(50) , Bank_Code
int );
ALTER TABLE Loan ADD FOREIGN KEY  ( Bank_Branch_Branch , Bank_Code )
REFERENCES Bank_Branch ( Branch , Bank_Code )  ON DELETE CASCADE ON
UPDATE CASCADE ;
CREATE TABLE A-C ( Customer_SSN int NOT NULL  , Account_AcctNo int NOT
NULL  , PRIMARY KEY  ( Customer_SSN , Account_AcctNo )  );
ALTER TABLE A-C ADD FOREIGN KEY  ( Customer_SSN ) REFERENCES Customer (
SSN )  ON DELETE SET NULL ON UPDATE CASCADE ;
ALTER TABLE A-C ADD FOREIGN KEY  ( Account_AcctNo ) REFERENCES Account
( AcctNo )  ON DELETE SET NULL ON UPDATE CASCADE ;
CREATE TABLE L-C ( Customer_SSN int NOT NULL  , Loan_LoanNo int NOT
NULL  , PRIMARY KEY  ( Customer_SSN , Loan_LoanNo )  );
ALTER TABLE L-C ADD FOREIGN KEY  ( Customer_SSN ) REFERENCES Customer (
SSN )  ON DELETE SET NULL ON UPDATE CASCADE ;
ALTER TABLE L-C ADD FOREIGN KEY  ( Loan_LoanNo ) REFERENCES Loan (
LoanNo )  ON DELETE SET NULL ON UPDATE CASCADE ;
```