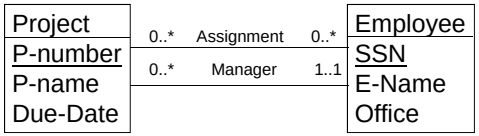
Week 5

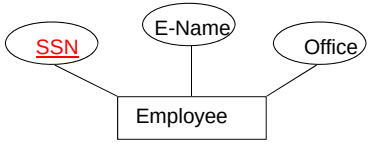
- Translating an ER Diagram to a Relational Schema
- Embedded SQL

Converting ER to Relational Schema



1. Translate each entity set into a table, with keys.

- Entity set:
  - can be represented as a table in the relational model
  - has a key … which becomes a key for the table

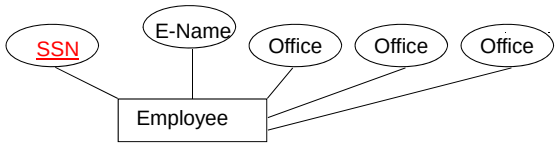


CREATE TABLE Employee
(SSN CHAR(11) NOT NULL,
E-Name CHAR(20),
Office INTEGER,
PRIMARY KEY (SSN))

Multi-valued Attribute

Didn't see this case when discussing ER diagrams

One or more values of same attribute for an entity



Most relational DBMSs do not allow multi-valued attributes.
2. Create a table for the multi-valued attribute.

How many offices can one employee have?

Just one

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

vs.

More than one

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name)
Office-Assignment(SSN, Office)

Sample Data

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, **Office**)

Just one

12	Smith	O-105
15	Wei	O-110

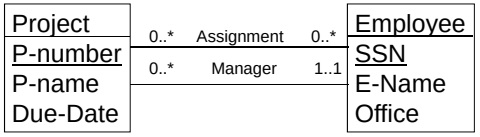
More than one

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name)

12	Smith
15	Wei

Office-Assignment(SSN, Office)

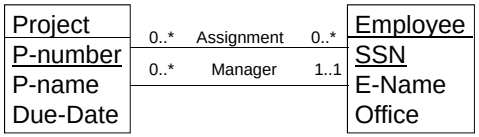
12	O-105
12	O-106
15	O-110



3. Translate each **many-to-many** relationship set into a table

What are the attributes and what is the key for Assignment?

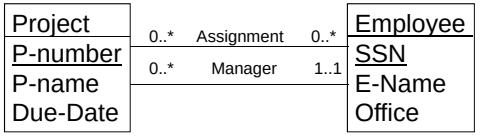
Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)



Answer: Assignment(P-Number, SSN)

P-Number is a foreign key for Project
SSN is a foreign key for Employee

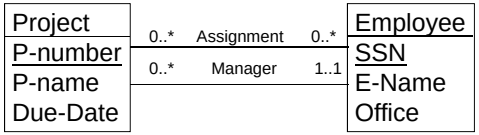
Project(P-Number, P-Due-Date)
Employee(SSN, E-Name, Office)



What should we do with each one-to-many relationship set?

Manager (?)

Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)

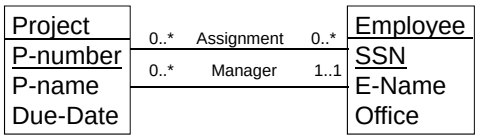


Project(P-number, P-name, Due-Date, MgrSSN)
Employee(SSN, E-Name, Office)

4. Create a foreign key for a 1-to-many relationship set.

MgrSSN is a foreign key (referencing the Employee relation)

value of Manager must match an SSN

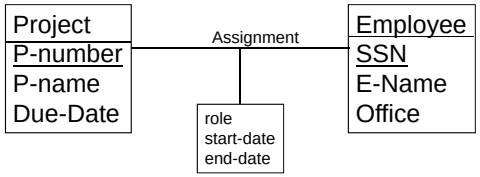


Project(P-number, P-name, Due-Date, MgrSSN)
Employee(SSN, E-Name, Office)

vs.

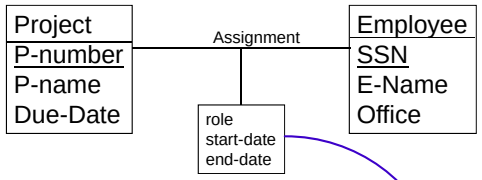
4. Or...Create a table for a 1-many relationship set.
Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)
Manager(P-number, SSN)
What are the tradeoffs between these two?

Note: P-number is the key for Manager



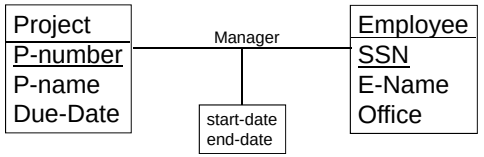
What do we do when a many-to-many relationship set has an attribute?

Assignment(P-number, SSN)
Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)



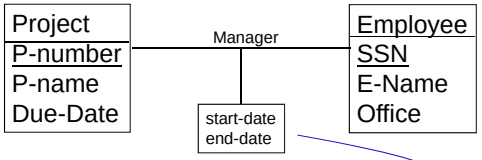
What do we do when a many-to-many relationship set has an attribute?

Assignment(P-number, SSN, role, start-date, end-date)
Project(P-number, P-name, Due-Date)
Employee(SSN, E-Name, Office)



What do we do when a 1-to-many relationship set has an attribute?

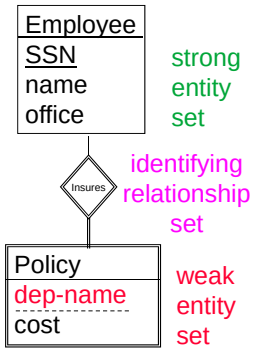
Project(P-number, P-name, Due-Date, MgrSSN)
Employee(SSN, E-Name, Office)



What do we do when a 1-to-many relationship set has an attribute?

Project(P-number, P-name, Due-Date, MgrSSN,
start-date, end-date)
Employee(SSN, E-Name, Office)

Weak Entity Sets



Translating Weak Entity Sets

- Weak entity sets and identifying relationship sets are translated into a single table. Must include key of strong entity set, as a foreign key.
- When the owner entity is deleted, all owned weak entities must also be deleted.

```
CREATE TABLE  Insurance_Policy (
  dep-name        CHAR(20),
  cost            REAL,
  ssn             CHAR(11) NOT NULL,
  PRIMARY KEY  (dep-name, ssn),

  FOREIGN KEY (ssn) REFERENCES Employee,
    ON DELETE CASCADE)
```


ER to Tables, Method 1

- Create table and choose key for each entity set; include single-valued attributes.
- Create table for each weak entity set; include single-valued attributes. Include key of owner as a foreign key in the weak entity. Set key as foreign key of owner plus local, partial key.
- For each 1:1 relationship set, add a foreign key to one of the entity sets involved in the relationship (a foreign key to the other entity in the relationship)*.
- For each 1:N relationship set, add a foreign key to the entity set on the N-side of the relationship (to reference the entity set on the 1-side of the relationship)*.

* Unless relationship set has attributes. If it does, create a new table for the relationship set.

ER to Tables, Method 1

- For each M:N relationship set, create a new table. Include a foreign key for each participant entity set, in the relationship set. The key for the new table is the set of all such foreign keys.
- For each multi-valued attribute, construct a separate table. Repeat the key for the entity in this new table. It will serve as a foreign key to the original table for the entity. The new key of the table will be the key of the entity plus the attribute.

This algorithm from Elmasri & Navathe, *Fundamentals of Database Systems*

ER to Tables, Method 2 (Book)

For regular entities and relationships (not weak/identifying)

- Create a table for each entity; include all attributes
- Create a table for each relationship; include attributes to represent the key for every participant in the relationship.

Not mentioned in the book (but important): Choose the key for each table.

- Easy for tables for entities
- Not as easy for tables that represent relationships

ER to Tables, Method 2 (Book)

For a weak entity and the identifying relationship

- Table for the weak entity must include the key from the strong entity in the identifying relationship (plus all attributes directly in the weak entity)
- Identifying relationships do NOT need tables.
- Other relationships involving the weak entity must use the concatenation of the partial key for the weak entity plus the key from the corresponding strong entity.

The key for the weak entity must be the partial key, plus the key of the corresponding strong entity.

ER to Tables, Method 2 (Book)

Then … combine tables

- If you have two or more tables with the same key … then combine them into one table.
- What's happening is that if the entity participates in a relationship where there is at most one of the other entity, the relationship can be represented as a foreign key.

Embedded SQL: What and Why?

- Embedded SQL allows data from a DBMS to be accessed programmatically
- Embedded SQL Programmers can:
  - Control how data is presented to users
  - Control what data is visible to users
  - Generate SQL dynamically based on user inputs

When are Queries Analyzed?

At compile time: Static SQL

At run time: Dynamic SQL

Static Case

- SQL commands are embedded in program with some kind of special delimiters
- Use a pre-processor that recognizes the SQL parts of the program
- Prepare once, execute many times
- Need a way to pass parameters to query
- Know schema of the result of a SELECT statement in advance

Dynamic Case

- Create a string at run time that represents the query
- Use function calls or methods to pass the string to the DBMS
  (no preprocessing of program)
- Analyze and execute the query each time
- Need a way to discover the schema of the result

Note: Some languages support both static and dynamic cases

Embedded SQL

- SQL commands can be called from within a host language (e.g., C/C++, Basic, .NET, PHP) program.
  - SQL statements can refer to host variables (including special variables used to return status).
  - Must include a statement to *connect* to the right database.
- SQL relations are (multi-) sets of records, with no *a priori* bound on the number of records. No such data structure in most languages.
  - SQL supports a mechanism called a *cursor* to handle this.

Cursors

- Can declare a cursor on a table or query statement (which generates a relation).
- Can *open* a cursor, and repeatedly *fetch* a row (then move the cursor), until all rows have been retrieved.
- May also be possible to modify/delete row pointed to by a cursor.
- Cursor can report conditions (e.g., end of rows)

Embedded SQL Implementations

- We will discuss four implementations of Embedded SQL
  - C (Pro*C from Oracle is an example)
  - Java (Using JDBC)
  - C# .NET (Using ADO to talk to MS Access)
  - Scripting (Using PHP & PostgreSQL)
    Illustrates database-backed web pages.

Example Table Schema

Imagine we are tracking products and categories for a retailer

```
Products(int ProductID,
  int CategoryID, String ProductName,
  currency UnitPrice)
Categories(int CategoryID,
  String CategoryName)
```


Cursor that gets the name and unit price of all beverages

```
DECLARE pinfo CURSOR FOR
  SELECT  P.ProductName, P.UnitPrice
  FROM  Products P, Categories C
  WHERE  C.CategoryName="Beverages"
      AND P.CategoryID=C.CategoryID
  ORDER BY  P.UnitPrice;

OPEN pinfo;

FETCH pinfo INTO :p-name, :p-price;
  (probably for each row)

CLOSE pinfo;
```


Embedding SQL in C: An Example

```
void ListProducts(short Max)
{
  char SQLSTATE[6];

  EXEC SQL BEGIN DECLARE SECTION
      char ProductName[20];
      float ProductPrice;
      short MaxPrice = Max;
  EXEC SQL END DECLARE SECTION
```

SQLSTATE holds the return value – can tell if more results, among other things

EXEC SQL denotes embedded SQL section – flag for preprocessor

DECLARE SECTION binds variables into SQL

Embedding SQL in C: (continued)

```
EXEC SQL DECLARE pinfo CURSOR FOR
    SELECT  P.ProductName, P.UnitPrice
    FROM  Products P, Categories C
    WHERE  C.CategoryName="Beverages"
          AND P.UnitPrice < :MaxPrice
          AND P.CategoryID=C.CategoryID
    ORDER BY  P.UnitPrice;
```

DECLARE pinfo CURSOR defines a name for this query for later use

SELECT P.ProductName … is our SQL that we want results on

< :MaxPrice - Note the use of a variable, defined earlier

Embedding SQL in C: (continued)

```
EXEC SQL OPEN pinfo;
EXEC SQL FETCH pinfo INTO :ProductName, :ProductPrice;
while (SQLSTATE != "02000") {
   printf("%s costs %f each\n", ProductName,
   ProductPrice);
   EXEC SQL FETCH pinfo INTO :ProductName, :ProductPrice;
};
EXEC SQL CLOSE pinfo;
```

OPEN pinfo – opens the query we are interested in

FETCH pinfo INTO – assigns data into our variables, for use in the output

ProductName – Use of our variable in and out of SQL EXEC

CLOSE pinfo – We're done with the cursor, free up its resources

Embedded SQL in Java

- Exposed through libraries called JDBC (Java DataBase Connectivity
- Using package java.sql.*
- All of the principles of cursors still apply
  - They are now encapsulated in object methods
- A Java Cursor is called a ResultSet Object
- Column names and positions are stored in a ResultSetMetaData object

Embedded SQL in Java

```
try {
Connection connect =
        DriverManager.getConnection("jdbc:mysql:
      www.mydomain.com:12543/mydb", username,
      password);

Statement st = connect.createStatement ();
}

catch (Exception e) {
      … exception thrown because connection failed …
}
```

Embedding SQL in Java: An Example

```
boolean status = st.execute ("SELECT * FROM
   Products");
if (status) {
   ResultSet rset = st.getResultSet ();
   ResultSetMetaData meta = rset.getMetaData ();
   … do stuff …}
else { … query did not return rows … }
```

rset – The Cursor Object (Recordset)

meta – the collection of columns and column positions for the records

else – the query may have been an insert, update, delete command

Metadata and ResultSet Objects

```
int nColumns = meta.getColumnCount();
String columnName = meta.getColumnLabel(i);
int columnWidth = meta.getColumnDisplaySize(i);
```

```
while (rset.next ()) {
      …
      Object val = rset.getObject (i);
            // getString, getBoolean, etc.
      rset.updateObject (i, obj);
            // updateString, updateBoolean, etc.
      …}
```


Embedding SQL in Java: An Example (continued)

```
rset.close ();
connect.close ();
```

- Don't forget to free up resources (result sets, connections)

Embedded SQL in .NET

- Exposed through the libraries System.Data.*
  - We'll focus on System.Data.OleDb
- Can be used in the Microsoft .NET framework (Basic, C#, Managed C++, JScript)
  - We'll use C#
- Object-Oriented
- A Cursor is a OleDbDataReader Object
- LINQ is another technology for connecting to data sources

Embedded SQL in .NET (C#)

```
OleDbCommand myCmd = new OleDbCommand
        ("SELECT P.ProductName, P.UnitPrice " +
        "FROM  Products P, Categories C " +
        "WHERE  P.CategoryID=C.CategoryID " +
        "AND C.CategoryName = \"Beverages\" " +
        "ORDER BY P.UnitPrice", myConn);

OleDbDataReader myRdr = myCmd.ExecuteReader();

Object name = myRdr.GetValue(0);

myRdr.Close();
```


Embedded SQL in .NET (C#): An Example

```
void ListProducts(int MaxPrice)
   OleDbCommand myCmd = new OleDbCommand();

try {
   myCmd.Connection = new OleDbConnection(
      "Provider=Microsoft.Jet.OLEDB.4.0;
       Data Source=products.mdb");
   myCmd.Connection.Open();
```

ListProducts(int MaxPrice) – definition of this subroutine

myCmd – The Command object to execute the SQL

new OleDb Connection – Opens the Access Database products.mdb

myCmd.Connection – Assigns a connection to the command

Embedded SQL in .NET (C#): An Example (continued)

```
myCmd.CommandText =  "SELECT P.ProductName, P.UnitPrice " +
            "FROM Products P, Categories C " +
            "WHERE P.UnitPrice < " + MaxPrice +
                  " AND C.CategoryName = \"Beverages\" " +
                  " AND P.CategoryID = C.CategoryID " +
            "ORDER BY P.UnitPrice";

OleDbDataReader myRdr = myCmd.ExecuteReader();
```

myCmd.CommandText – Assign the desired SQL query

myRdr = myCmd.ExecuteReader() – Open a cursor on the SQL

"P.UnitPrice < " + MaxPrice - Use a variable defined in C#

Embedded SQL in .NET (C#): An Example (continued)

```
   while (myRdr.Read()) {
      Console.WriteLine(myRdr.GetString(0) + " costs $" +
            myRdr.GetDecimal(1) + " each“);
      }
} //end try
```

while (myRdr.Read()) – Read gets the next record. If it's after the last record, Read() returns False

myRdr.Get***** - OleDbDataReader provides many Get functions to retrieve different data types. GetFieldType() returns the types for each attribute, if they are not already known

Console.WriteLine – Write to the command window

Embedded SQL in .NET (C#): An Example (continued)

```
catch (Exception ex) {
    Console.WriteLine(ex.Message);
}
```

catch – catches any exception that is thrown during execution

Message is a member of the Exception object.

Embedded SQL using PostgreSQL and PHP

- Scripting languages such as Perl, Python, and PHP have database support
  - We'll talk some about PHP next Thursday
- To work with a PostgreSQL database in a scripting language, you must use a set of functions designed to communicate with PostgreSQL
- PHP includes the PostgreSQL functions - you do not need an additional library

Other Embedded SQL Solutions

- ODBC – Open Database Connectivity
  - Old standard, proposed by Microsoft but driven by the database community
  - Many vendors, including Oracle, make ODBC drivers available
- Haskell
  - HaskellDB (currently uses ADO)
  - Cursors are first-class objects (each attribute is a type)