

Week 5

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

Exam Coverage

Assigned readings from Chapters 1-5

Main topics

- Relational model, keys, foreign keys
- SQL
 - SELECT ... FROM ... WHERE ...
 - GROUP BY, HAVING
 - JOINS
 - Subqueries
- Relational Algebra
- Views
- Use cases, ER diagrams and translation to relations
- Embedded SQL

What's OK and Not OK for Exam

OK:

- Course text
- SQL manual
- Lecture notes, quizzes, HWs, class handouts
- Foreign language dictionary

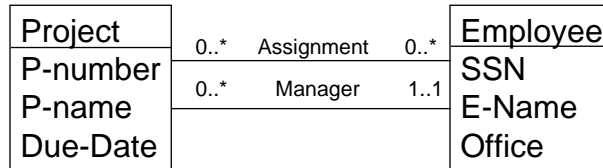
Not allowed:

- Other books
- Materials not from this course offering
- Computers, wireless devices

HW 4

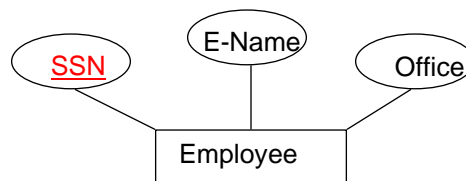
We will try to get a solution set posted for
HW 4 by next Monday.

Example ER Diagram



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

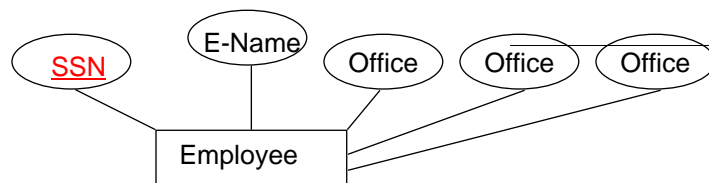


```
CREATE TABLE Employee
(SSN CHAR(11),
E-Name CHAR(20),
Office INTEGER,
PRIMARY KEY (SSN))
```

Multi-valued Attribute

Didn't see this case when discussing ER diagrams

One or more values of same attribute for an entity



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 7
Lecture 5

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)

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 8
Lecture 5

Sample Data

Project (P-number, P-name, Due-Date)

Employee (SSN, E-Name, Office)

Just one

12 Smith O-105

15 Wei O-110

20 Jones O-112

Project (P-number, P-name, Due-Date)

Employee (SSN, E-Name)

More than one

12 Smith

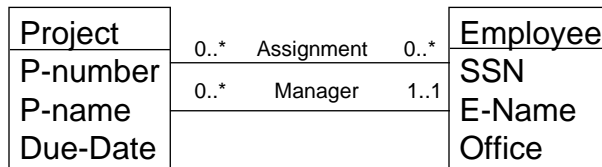
15 Wei

Office-Assignment (SSN, Office)

12 O-105

12 O-106

15 O-110



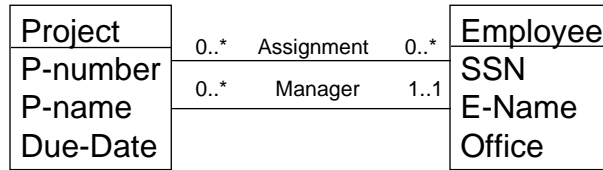
3. Translate each **many-to-many** relationship set into a table

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

Assignment (?)

Project (P-number, P-name, Due-Date)

Employee (SSN, E-Name, Office)



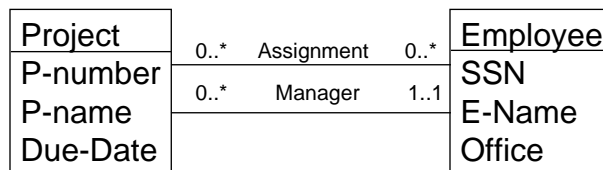
Answer: Assignment (P-Number, SSN)

P-Number is a foreign key for Project

SSN is a foreign key for Employee

Project (P-Number, P-Name, P-Due-Date)

Employee (SSN, E-Name, Office)

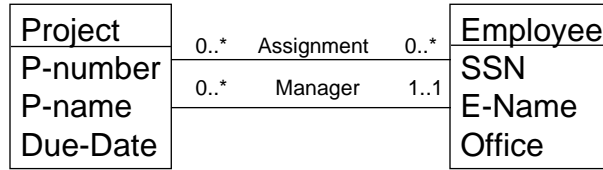


What should we do with each **one-to-many** relationship set?

Manager (?)

Project (P-number, P-name, Due-Date)

Employee (SSN, E-Name, Office)

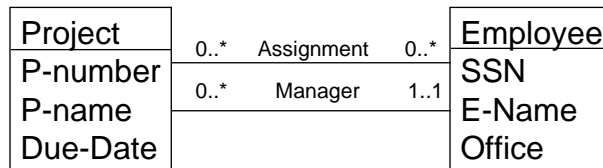


Project (P-number, P-name, Due-Date, **Manager**)
 Employee (SSN, E-Name, Office)

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

Manager is a foreign key (referencing the Employee relation)

value of Manager must match an SSN



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

vs.

4. Or...Create a table for a 1-many relationship set.

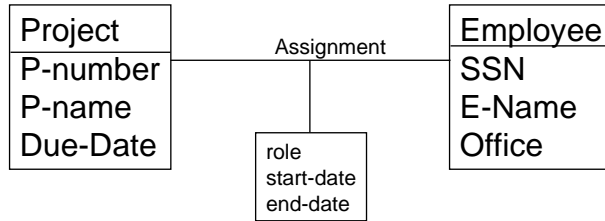
Project (P-number, P-name, Due-Date)

Employee (SSN, E-Name, Office)

Manager (P-number, SSN)

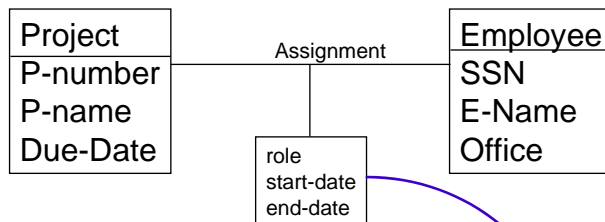
What are the tradeoffs between these two?

Note:
 P-number
 is the key
 for Manager



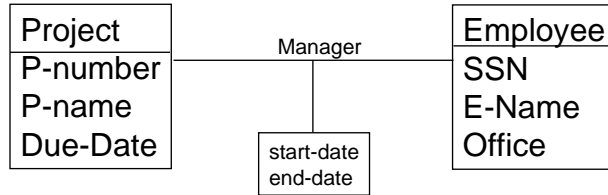
What do we do when a many-to-many relationship set has an attribute?

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



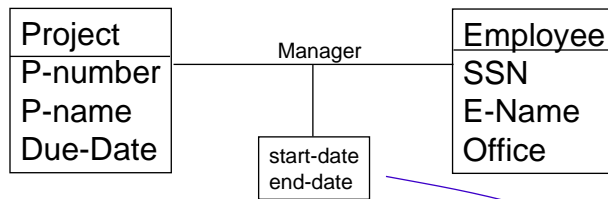
What do we do when a many-to-many relationship set has an attribute?

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, Manager)
 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, Manager, start-date, end-date)
 Employee (SSN, E-Name, Office)

Participation Constraints in SQL

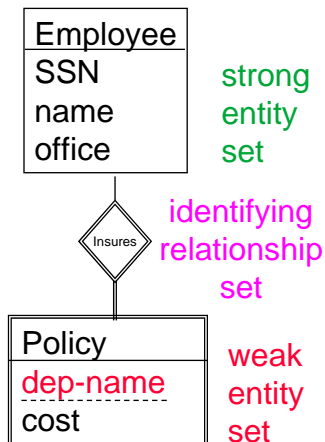
- We can require any table to be in a binary relationship using a foreign key which is required to be NOT NULL (but little else without resorting to CHECK constraints)

```
CREATE TABLE Department (
  d-code          INTEGER,
  d-name          CHAR(20),
  manager-ssn    CHAR(9) NOT NULL,
  since          DATE,
  PRIMARY KEY (d-code),
  FOREIGN KEY (manager-ssn) REFERENCES Employee,
  ON DELETE NO ACTION)
```

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 19
Lecture 5

Weak Entity Sets



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 20
Lecture 5

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)
```

Note ERDs and UML Diagrams can be at two levels:

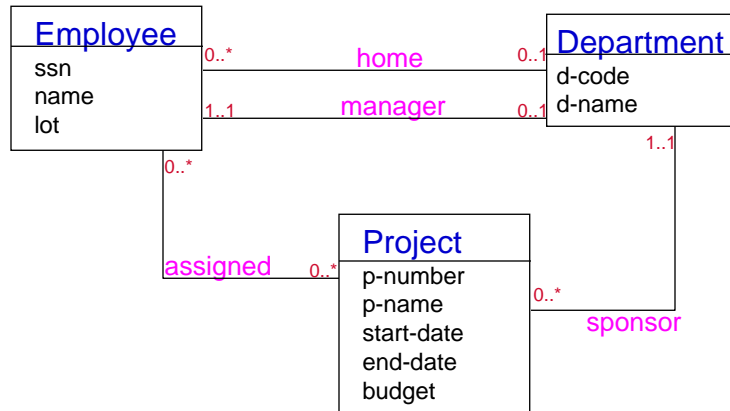
the ERD/UML level

and

the Relational Table level.

The difference is primarily with the many-to-many relationship sets.

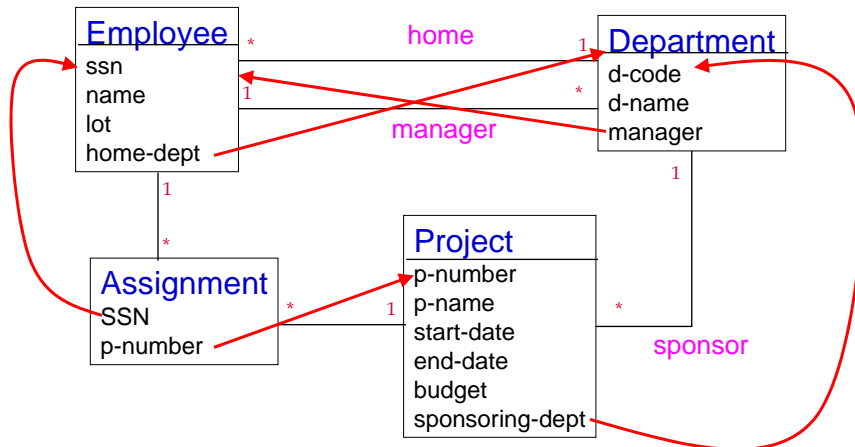
Entity-Relationship Diagram



CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 23
Lecture 5

Equivalent Relational Schema



Notice that the relationship sets shown in this diagram aren't really needed. *foreign keys* reference other tables.

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 24
Lecture 5

Translation Steps: ER to Tables

- 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.

Translation Steps: ER to Tables

- 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 both the key for this table as well as a foreign key to the original table for the entity.

This algorithm from Elmasri/Navathe, p. 174.

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 preferences

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)
- 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 relation or query statement (which generates a relation).
- Can *open* a cursor, and repeatedly *fetch* a tuple then *move* the cursor, until all tuples have been retrieved.
- May also be possible to modify/delete tuple pointed to by a cursor.
- Must be able to report data-generated errors.

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)
Will see an example application in this form
- Comparing implementations
 - How do each address the requirements of embedded SQL?
 - ~~What more do they offer?~~

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 33
Lecture 5

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)
```

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 34
Lecture 5

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;

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 35
Lecture 5

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

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 36
Lecture 5

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 Recordset Object
- Column names and positions are stored in a `RecordSetMetaData` object

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 39
Lecture 5

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 ...
}
```

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 40
Lecture 5

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 RecordSet Objects

```
int nColumns = metadata.getColumnCount ();
String columnName = metadata.getColumnLabel (i);
int columnWidth = metadata.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
- Though usable in .NET, it is included only with Visual Studio.NET

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();
```

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 45
Lecture 5

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

CS386 Introduction to Database Systems, © Lois Delcambre, David Maier 1999-2006
Some slides adapted from R. Ramakrishnan, with permission

Slide 46
Lecture 5

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 focus on 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)