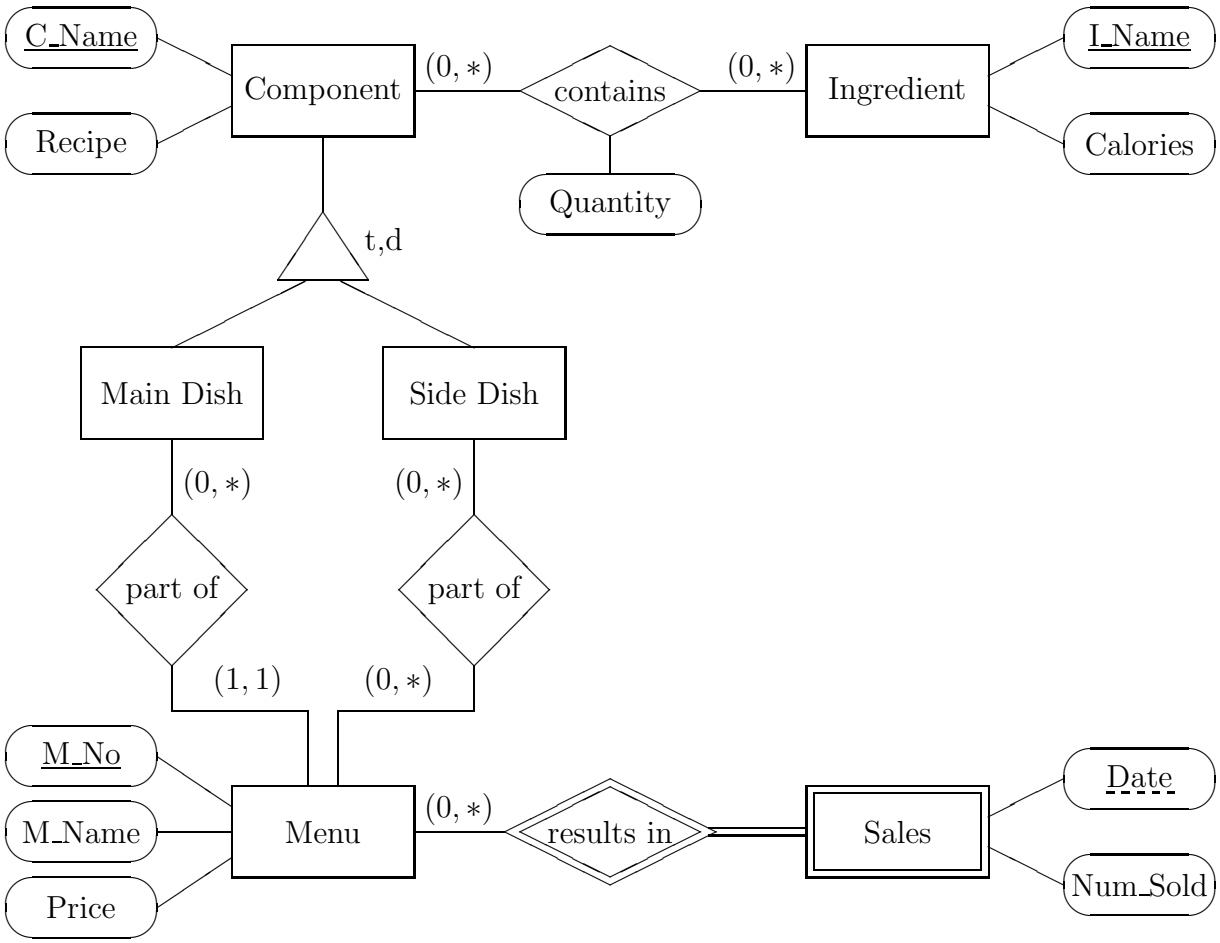
Dr. Stefan Brass
School of Information Sciences
University of Pittsburgh

February 18, 2000

INFSCI 2710 "Database Management" — Solution to Example Midterm Exam II —

Exercise 1 (ER Design) — 12 Points

This exercise was a bit difficult because the application is from German canteens/cafeterias for students. The American system is different. I will try to use other examples in future which are better known.



You do not need to use subclasses for main dishes and side dishes. If you use an attribute "Type", you must add a constraint that every Menu contains only one main dish.

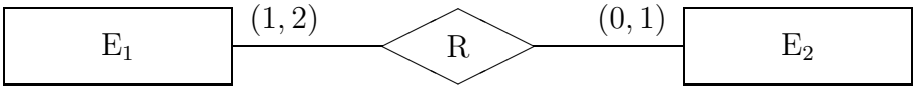
You do not need to use a weak entity for the sales. You could use an entity for Date and put "Num_Sold" as an attribute to the relationship "offered_on" between Menu and Date.

Exercise 2 (Comparison of ER-Schemas) 3 Points

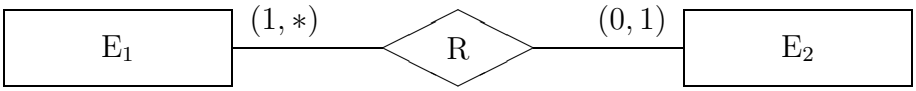
The two entity-relationship diagrams are equivalent (can store exactly the same information). Since the relationships "from" and "to" have the cardinality "(1,1)", every flight must participate in these relationships exactly once. But then the attributes "Departure" and "Arrival" are uniquely defined for every flight and can as well be put as attributes directly to "Flight".

Exercise 3 (Cardinalities) 2 Points

a) The strongest (most restrictive) cardinalities you can define such that the given state is valid, are:

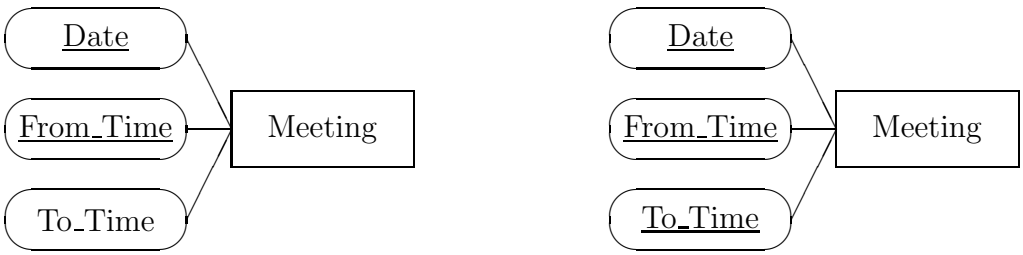


b) There are many possibilities to weaken the cardinalities. One example would be:



Exercise 4 (Keys) 3 Points

The left key is more restrictive, because the key attributes are a subset of the right key.



For example, the following state would be valid with respect to the right key, but invalid with respect to the left key:

Meeting		
Date	From	To
2/16/00	10:00	11:00
2/16/00	10:00	12:00

Here, the state for the ER-schema is shown as the table which would result from the translation into the relational model.

Exercise 5 (Relational Algebra) — 15 Points

a) Give all information about flower bouquets which cost over $50.

$$\sigma_{\mathsf{Price} > 50}(\mathsf{Flowers})$$

b) Print the recipient name and address of all orders to be delivered in Pittsburgh after 10/07/99. You can compare date values by the standard operators <, >, <=, >=.

$$\pi_{\mathsf{RName},\,\mathsf{RAddress}}\Big(\sigma_{\mathsf{RCity}=\text{'Pittsburgh'} \wedge \mathsf{Date} > \text{'10/07/99'}}(\mathsf{Order})\Big)$$

c) Print the name and city of all customers who ordered product number 12.

$$\pi_{\mathsf{CName},\,\mathsf{CCity}}\Big(\sigma_{\mathsf{Prod}=12}(\mathsf{Order} \bowtie \mathsf{Customer})\Big)$$

d) Print the numbers of all orders ("Ord") such that the person who ordered the flowers lives in the same city as the recipient of the flowers.

$$\pi_{\mathsf{Ord}}\Big(\sigma_{\mathsf{RCity} = \mathsf{CCity}}(\mathsf{Order} \bowtie \mathsf{Customer})\Big)$$

e) Print the cities and states to which orders were mailed via FedEx.

$$\pi_{\mathsf{RCity},\,\mathsf{RState}}(\mathsf{Order} \bowtie \mathsf{Mailed})$$

Exercise 6 (Relational Algebra Errors) — 2 Points

This query is wrong because "Description" is no attribute in "Customer" or "Order", and thus also the result of the join has no such column:

$$\pi_{\mathsf{Description}}(\mathsf{Customer} \bowtie \mathsf{Order})$$

Exercise 7 (SQL) — 9 Points

a) Give all information about flower bouquets which contain the substring "roses" in their description. (You must use the `LIKE` operator of SQL for this.)

```
SELECT *
FROM   Flowers
WHERE  Description LIKE '%roses%'
```

b) Print the recipient name and address of all orders to be delivered in Pittsburgh or Philadelphia on 10/07/99. (Note that the date restriction applies to deliveries in both cities.) Date values are normally written in Oracle as `'10-JUL-99'`.

```
SELECT RName, RAddress
FROM   Order
WHERE  (RCity = 'Pittsburgh' OR RCity = 'Philadelphia')
AND    Date = '10-JUL-99'
```

c) Print the order number, recipient name, city and state of all orders for flowers which cost over $50.

```
SELECT Ord, RName, RCity, RState
FROM   Order, Flowers
WHERE  Order.Prod = Flowers.Prod
AND    Price > 50
```

Exercise 8 (SQL Errors) — 6 Points

a) Single quotes must be used for strings.

b) "`Ord`" in the `SELECT`-clause is ambigious: You must write either "`Order.Ord`" or "`Subcontracted.Ord`". In addition, the table name is "`Order`", and not "`Orders`" (unintended typing error). Naming one of the two errors was sufficient, naming both gave extra credit.

c) The `FROM`-clause is missing. It is mandatory. Every SQL query must contain `SELECT` and `FROM`.