L17: Case Study

Mongo's Kitchen

Dr Kieran T. Herley

Semester One, 2023-24

School of Computer Science & Information Technology
University College Cork

Summary

Development of ER model for database for Mongo's Takeaway

Mongo's Takeaway

- Mongo's famous takeaway wants DB-based order-submission website
- Needs DB backend to house data to support this

- Model:
  - Customer submit order via fancy form
  - Receives confirmation with order num and price
  - Collects completed order in person later

Mongo's Menu

code	item	price
mo1	Cheeseburger	€6.00
mo2	Chips	€1.50
mo3	Mushy Peas	€2.00
mo4	Sauerkraut	€2.00
mo5	Pizza	€7:50
mo6	Kebab	€5.25
mo7	Spam	€3.00
mo8	Lobster Thermidor	€30.00

Order Details

For each order we need to keep track of

- customer's details
- items ordered and quantity (and price)
- status:

  `received -> posted -> completed -> collected`

Sample Queries

- DB should support order processing such as
  - List items on menu and prices
  - Calculate price of an order
  - Generate list of active orders

Sample Queries

- DB should support order processing such as
  - List items on menu and prices
  - Calculate price of an order
  - Generate list of active orders
- Should also help production of business summary info.
  - Determine total sales for last month
  - Determine max, min, average wating time for each day
  - List customers in decreasing order of total sales for the month
  - List customers who places an order but never collected it

Towards a DB Design

Entity Sets

- customers: names, addresses etc.
- menu_items: items on offer, descriptions, prices
- orders: order number

Towards a DB Design

Entity Sets

- customers: names, addresses etc.
- menu_items: items on offer, descriptions, prices
- orders: order number
- staff:
  - name and staff number
  - (To keep track of who handles order in case of problems)

Towards a DB Design cont'd

Relationships

- placed_by: which customers placed which orders

Towards a DB Design cont'd

Relationships

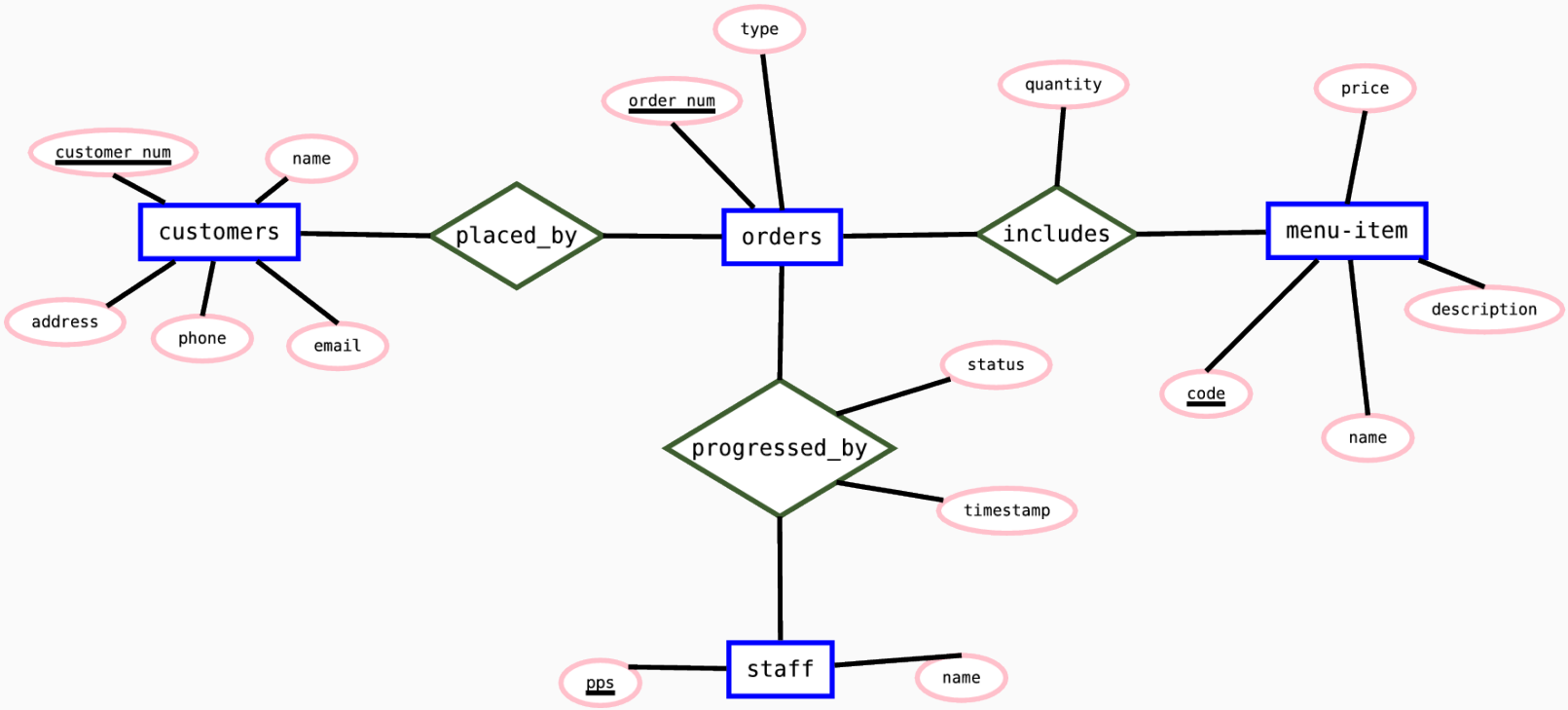
- placed_by: which customers placed which orders
- includes:
  - which items are in which orders
  - attribute: quantity of item in this order

Towards a DB Design cont'd

Relationships

- placed_by: which customers placed which orders
- includes:
  - which items are in which orders
  - attribute: quantity of item in this order
- progressed_by:
  - which staff members processed the order
  - attributes: order status and time-stamps
  - (tracks order progress and times)

Complete ER Diagram



DB Schema

```
customers(customer_num, name, address, email, phone)
orders(order_num, order_type)
menu_items(code, name, description, price)
staff(pps, name)

placed_by(customer_num, order_num)
progressed_by(order_num, staff_num, status, time_of_change)
includes(order_num, menu_code, quantity)
```

Note: no weak entity types in this design

Typical DB Interactions

- List the available menu items and prices
- Determine price of an order
- List all outstanding orders order of urgency (oldest first)
- List total sales for last month by menu item
- List customers in order of the numbers of orders placed last month.

Displaying The Menu

Task List the available menu items and prices

Query

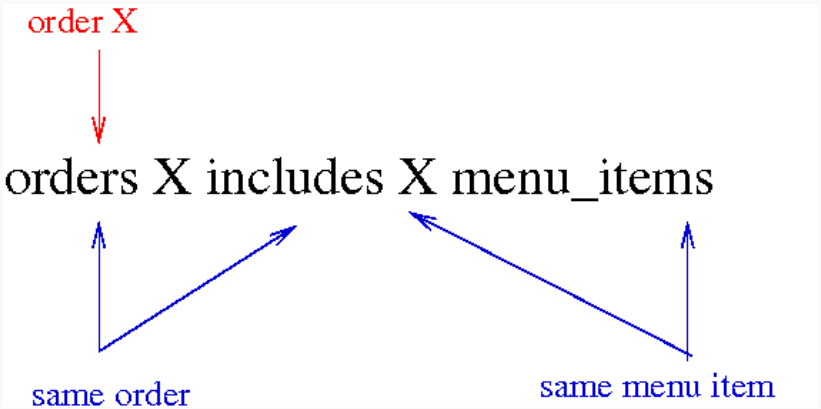
```
SELECT code, name, description, price
FROM menu_items;
```

Notes (Embedded in PHP script, which pretty-prints result as a form using HTML/CSS)

Pricing An Order

Task Determine the price of an order (given its order number, say X)

Idea

- 

- One row in filtered combo table per order item

Pricing cont'd

Query

```
SELECT SUM(m.price*i.quantity) AS 'Total'
FROM
    orders  AS o JOIN
    includes  AS i JOIN
    menu_items AS m
    ON
      o.order_num = i.order_num AND
      i.menu_code = m.code
WHERE o.order_num = X;
```

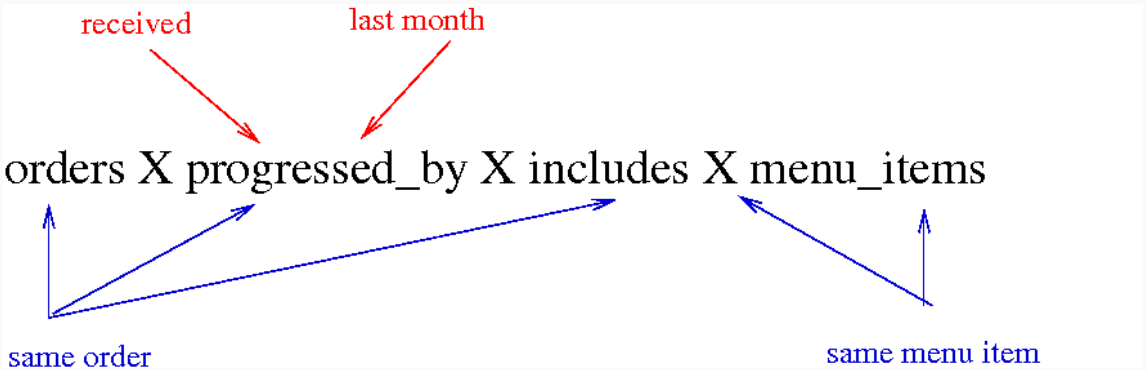
Notes

- Combines price and quantity for each "line" of order

Generate Sales Summary

Task List total sales for each menu item for last month

Idea

- 

- Group order items for same menu item together

Sales Summary cont'd

Query

```
SELECT m.name, SUM(i.quantity), SUM(m.price*i.quantity)
FROM
    orders  AS o JOIN
    progressed_by  AS pr JOIN
    includes  AS i JOIN
    menu_items AS m
    ON
      o.order_num = i.order_num AND
      pr.order_num = o.order_num AND
      i.menu_code = m.code
WHERE
    pr.`status` = 'received' AND
    pr.time_of_change BETWEEN '2012-11-01 00:00:00' AND
                              '2012-11-30 23:59:59'
GROUP BY m.code;
```

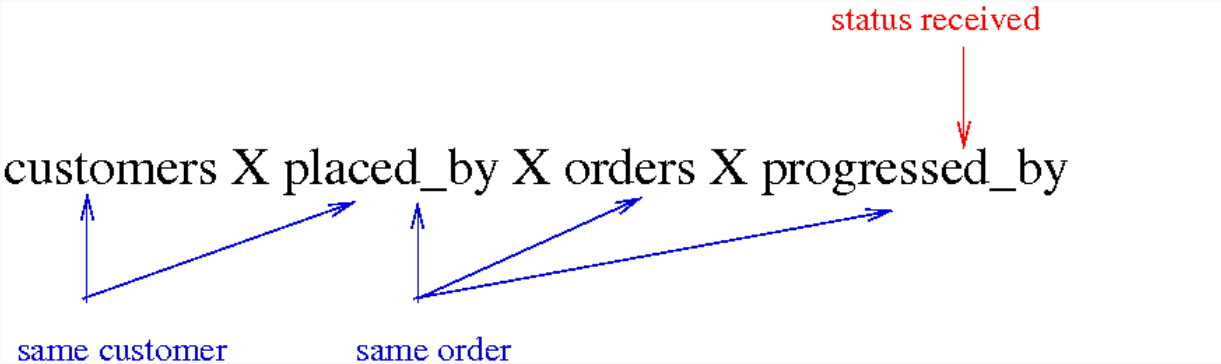
Notes

- Note DATETIME values combine dates and times: useful for "timestamps"

Listing “Active” Orders

Task List customers and order numbers of all “active” (uncompleted) orders, oldest first

Idea



Listing "Active" Orders cont'd

Query

```
SELECT pr.time_of_change, c.name, o.order_num
FROM
    customers AS c JOIN
    placed_by AS pl JOIN
    orders AS o JOIN
    progressed_by AS pr
    ON
      c.customer_num = pl.customer_num AND
      pl.order_num = o.order_num AND
      o.order_num = pr.order_num
WHERE pr.status = 'received'
ORDER BY pr.time_of_change;
```

Notes

- Includes completed orders!
- Refine to exclude orders that have been completed

Listing "Active" Orders cont'd

```
SELECT pr.time_of_change, c.name, o.order_num
FROM
    customers AS c JOIN
    placed_by  AS pl JOIN
    orders  AS o JOIN
    progressed_by  AS pr
    ON
      c.customer_num = pl.customer_num AND
      pl.order_num = o.order_num AND
      o.order_num = pr.order_num
WHERE
    pr.status   = 'received' AND
    NOT EXISTS                                  -- True only if
    (   SELECT * FROM progressed_by AS pr2 -- order was
      WHERE                                     -- completed
      pr.order_num = pr2.order_num AND
          pr2.status  = 'completed'             --
    )
ORDER BY pr.time_of_change;
```


- Subquery returns nin-empty result if `progressed_by` has row with appropriate order number and status

Notes and Acknowledgements

The restaurant image is a still from Monty Python's "Dirty Fork" sketch.