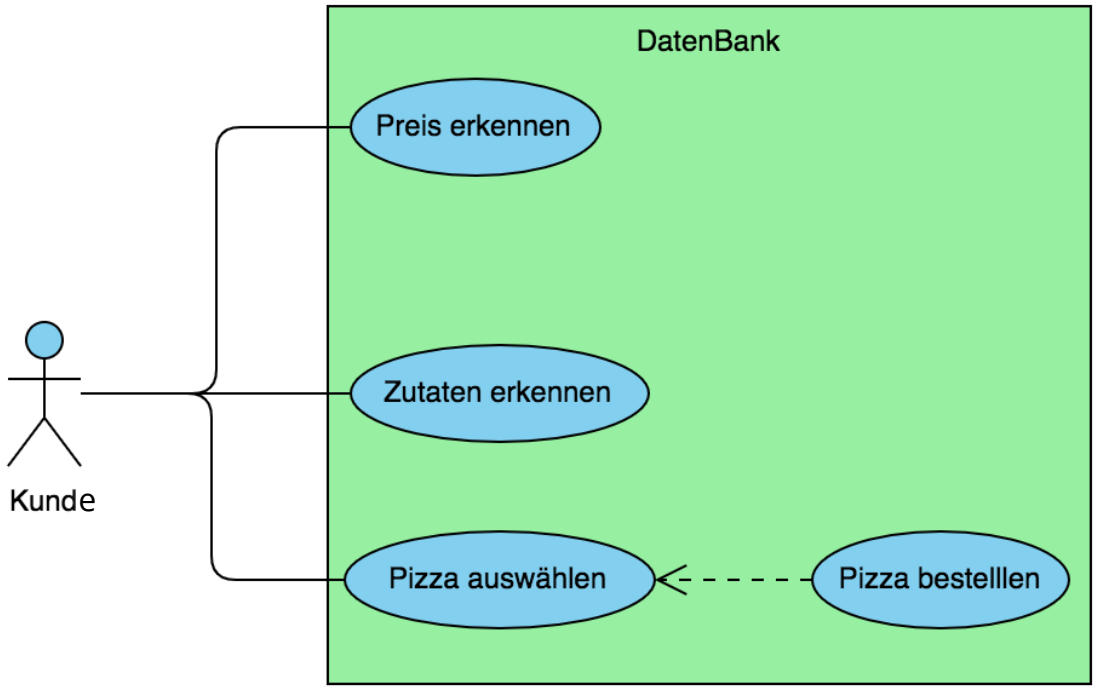
Pizzagedon

Peter Marc Bienas

Diese Datenbank speichert verschiedene Arten von Pizza und kann deren Inhalt, Name und Preis angeben. Jede Bestellung besteht aus einem Kunden und einer Pizza

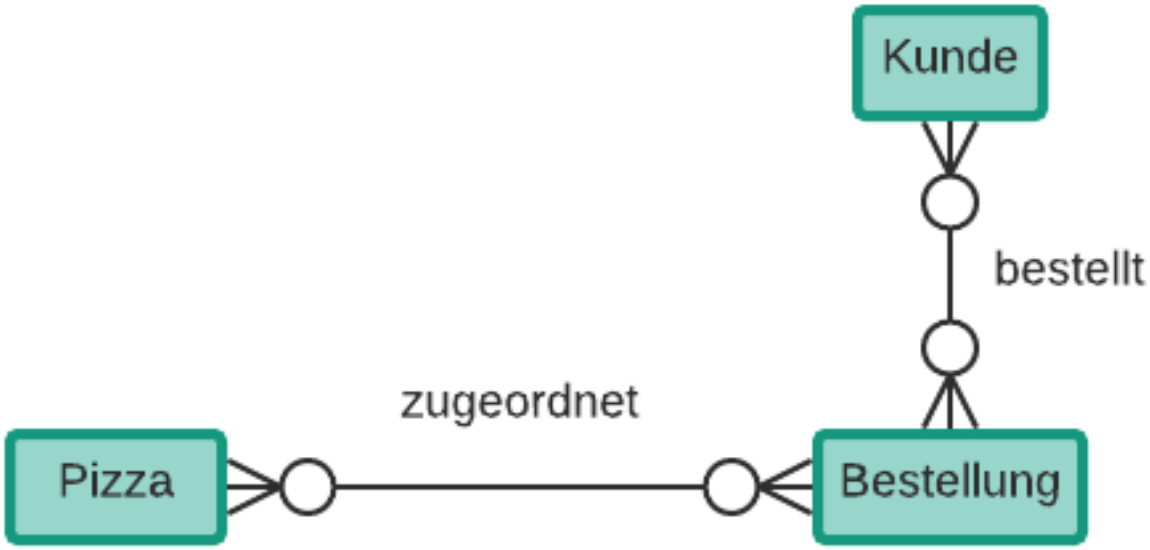


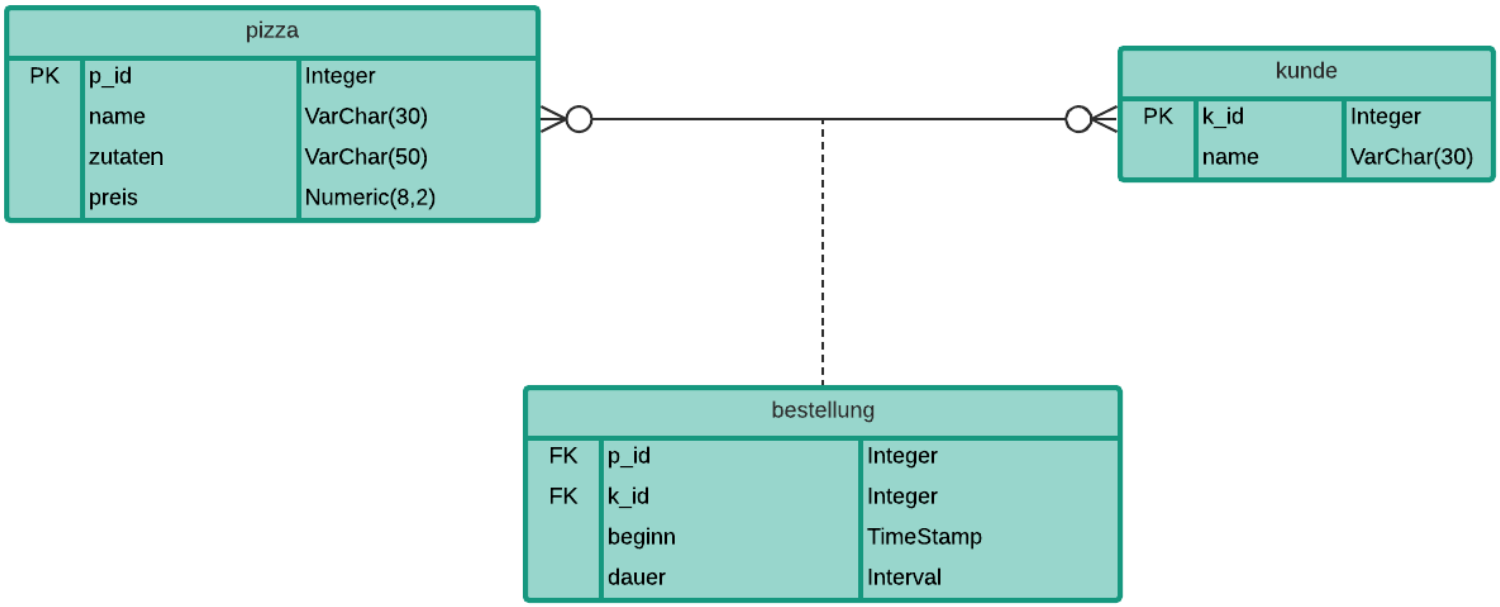
Preis erkennen:
Akteur: Kunde
Der Kunde kann erkennen Wie viel jede Pizza Kostet und kann darauf seine Kaufentscheidung treffen.

Zutaten erkennen:
Akteur: Kunde
Im Falle das der Kunde Allergisch ist, kann er erkennen was sich in jeder Pizza befindet und wird somit im stande sein allergische Unfälle zu umgehen.

Pizza auswählen:
Akteur: Kunde
Der Kunde ist sich sicher und wählt die Pizza die er haben will, daraufhin wird die Zubereitung der pizza gestartet.

Pizza bestellen:
Akteur: Kunde
Durch das auswählen einer Pizza wird sie gleich bestellt.


Kunde
bestellt
zugeordnet
Pizza
Bestellung


pizza
PK
p_id
Integer
name
VarChar(30)
zutaten
VarChar(50)
preis
Numeric(8,2)
kunde
PK
k_id
Integer
name
VarChar(30)
bestellung
FK
p_id
Integer
FK
k_id
Integer
beginn
TimeStamp
dauer
Interval

```
DOMAIN D_NAME =VARCHAR(30)
CHECK(VALUE äöü ~*'\A[\wäöüß_.@$+\-/()\s]+\Z')
DOMAIN D_ZUTATEN =VARCHAR(100)
CHECK(VALUE äöü ~*'\A[\wäöüß_.@$+\-/()\s]+\Z')
DOMAIN D_PREIS =NUMERIC(8,2)
CHECK(VALUE ~'\A[A-Z][0-9][.][0-9][0-9][a-z]?\Z')
DOMAIN D_BEGINN = TIMESTAMP
CHECK(VALUE ~'\A[0-9][0-9\-/()\s]*\Z')
DOMAIN D_DAUER =INTERVAL
CHECK(VALUE ~'\A[0-9][0-9\-/()\s]*\Z')

pizza:
p_id INTEGER, name D_NAME, zutaten D_ZUTATEN, preis D_PREIS
PK: p_id

kunde:
k_id INTEGER, name D_NAME
PK: k_id

bestellung:
p_id, k_id, beginn D_BEGINN, dauer D_DAUER
FK: p_id->pizza(p_id), k_id->kunde(k_id)
```

```
DROP TABLE IF EXISTS pizza CASCADE;
DROP TABLE IF EXISTS kunde CASCADE;
DROP TABLE IF EXISTS bestellung CASCADE;

CREATE TABLE pizza (
   p_id INTEGER NOT NULL PRIMARY KEY,
   name VARCHAR(30),
   zutaten VARCHAR(50),
   preis NUMERIC(8,2)
);

CREATE TABLE kunde (
   k_id INTEGER NOT NULL PRIMARY KEY,
   name VARCHAR(30)
);

CREATE TABLE bestellung (
   p_id INTEGER NOT NULL,
   k_id INTEGER NOT NULL,
   beginn TIMESTAMP,
   dauer INTERVAL,
   FOREIGN KEY (k_id) REFERENCES kunde (k_id),
   FOREIGN KEY (p_id) REFERENCES pizza (p_id)
);
```

```
INSERT INTO pizza(p_id,name ,zutaten ,preis) VALUES
   (1, 'Peperoni', 'Tomato, Salami, Peperoni', 4.50),
   (2, 'Cheesagedon', 'Cheese', 4.00),
   (3, 'Empty Pizza', 'NULL', 8.00),
   (4, 'Sea Food Pizza', 'Tomato, Fish, Onion', 5.00)
;

INSERT INTO kunde(k_id, name) VALUES
   (1, 'Tony'),
   (2, 'Peter')
;

INSERT INTO bestellung(p_id, k_id, beginn, dauer) VALUES
   (2,1, TIMESTAMP '2019-08-29 14:00', INTERVAL '20 minutes'),
   (3,2, TIMESTAMP '2019-08-29 15:00', INTERVAL '5 minutes')
;
```

```
--VIEWS
--legt view an
CREATE VIEW pizza_info(name, zutaten, preis)
AS
SELECT p.name, p.zutaten, p.preis
FROM pizza p

--sucht view ab nach tomato
SELECT *
FROM pizza_info
where zutaten LIKE 'Tomato%'

--sucht die billigste pizza mit View
SELECT *
FROM pizza_info
ORDER BY preis ASC
LIMIT 1

--sucht nach einer speziellen pizza
SELECT *
FROM pizza_info
WHERE name LIKE 'Empty Pizza%'

--wie viele pizzen hat die view
SELECT COUNT(*) AS anzahl
FROM pizza_info
```

```
--NULLWERTE
--alle pizzen die nicht bestellt werden, werden dargestellt
SELECT *
FROM pizza p
LEFT JOIN bestellung b USING (p_id)
LEFT JOIN kunde k USING (k_id)
EXCEPT
SELECT *
FROM pizza p
LEFT JOIN bestellung b USING (p_id)
LEFT JOIN kunde k USING (k_id)
WHERE k.name IS NOT NULL

--die leere pizza wird "belegt"
Update pizza
set zutaten= 'Luft und Liebe'
where zutaten = 'NULL' and p_id = '3'

--alle pizzen die nicht Null sind wurden bestellt
SELECT p.name
FROM pizza p
LEFT JOIN bestellung b USING (p_id)
LEFT JOIN kunde k USING (k_id)
WHERE k.k_id IS NOT NULL
```

```
--VOLLTEXTSUCHE
--sucht alle pizzen heraus die peperoni haben
SELECT name
FROM pizza
WHERE CONTAINS(zutaten, 'Peperoni')

--gibt aus welche pizzen essbar sind für jemand der allergisch gegen
nichts ist
SELECT name
FROM pizza
WHERE CONTAINS(zutaten, 'NULL')

-- zeigt alle zutaten dar die den buchstaben o beinhalten
SELECT zutaten, name
FROM pizza
WHERE CONTAINS(zutaten, '_o%')
```

```
--AGGREGATION
--zeigt pizzen und wie nahe deren preis am durchschnitt liegt
select avg(p1.preis) as durchschnitt, p2.preis,p2.name
from pizza p1, pizza p2
group by p2.name, p2.preis
order by p2.preis, p2.name

--zeigt die teuerste pizza mit 2.50$ rabat
SELECT sub.preis AS ueblich, (sub.preis-2.50) AS reduziert, sub.name
FROM  (
      SELECT MAX(preis), name, preis
      FROM pizza
      GROUP BY name,preis
      LIMIT 1
)sub

--pizzen die unter dem durchschnitts preis liegen sind kostenlose
SELECT sub.name AS for_free
FROM  (
      SELECT AVG(p1.preis) AS average, p2.name, p2.preis
      FROM pizza p1, pizza p2
      GROUP BY p2.name, p2.preis
      )sub
WHERE sub.preis < sub.average
```

```
--subqueries
--subquery with benutzt um alle daten von peter zu bekommen
SELECT sub.*
FROM  (
      SELECT *
      FROM kunde
      WHERE name LIKE 'Peter%'
      ) sub

-- wählt alle pizzen mit tomato die billiger sind als 5.00 von view
SELECT sub.*
FROM  (
      SELECT *
      FROM pizza_info
      WHERE zutaten LIKE 'Tomato%'
      ) sub
WHERE preis < 5.00

--kreiere eine imaginäre view mit der man daten von tony bekommt
SELECT sub.*
FROM  (
      --versetzt select abfrage zwecks platzmangels
      SELECT k.name as kunde,
      p.name as bestellt,
      b.dauer as ist_fertig_in
      FROM bestellung b
      left join pizza p using (p_id)
      left join kunde k using (k_id)
      ) sub
where sub.kunde LIKE 'Tony%'

--bekomme alle pizza namen über einen umständlichen weg
Select subC.*
from (
      select subB.name
      from  (
            select subA.name, subA.zutaten
            from  (
                  select name ,zutaten , preis
                  from pizza
                  )subA
            )subB
     )subC

-- wie viele pizzen wurden pro typ bestellt
SELECT sub.*
FROM  (
      SELECT COUNT(*) AS anzahl, p.name AS pizza
      FROM bestellung
      JOIN pizza p USING(p_id)
      JOIN kunde k USING(k_id)
      GROUP BY (pizza)
      )sub
```