Restaurant Carryout System

Cameron Garratt, Colin Wahl, Nicolas Burgess,
Vivek Patel, Emma Fick

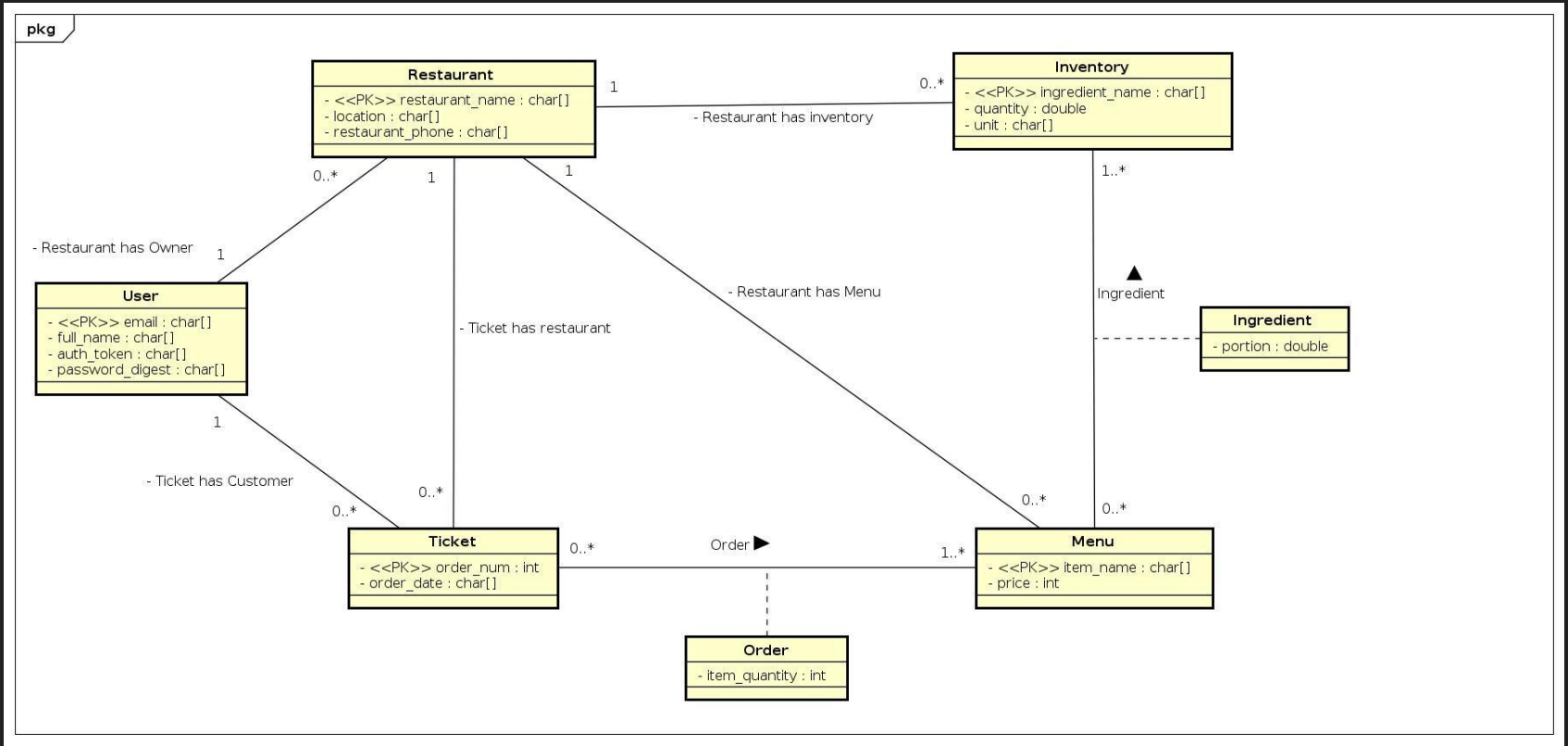
We Make It, You Take It!

A carryout and inventory management system all in one!

The restaurant carryout system we have designed allows for a restaurant manager to create multiple restaurants, menu items, and recipes. The web app also allows for you to manage the restaurant's current inventory.

Once a restaurant has been created a user can order any menu item from any of the restaurants you have created.

UML


pkg
Restaurant
- <<PK>> restaurant_name : char[]
- location : char[]
- restaurant_phone : char[]
Inventory
- <<PK>> ingredient_name : char[]
- quantity : double
- unit : char[]
1
0..*
- Restaurant has inventory
0..*
1
1
1..*
- Restaurant has Owner
1
User
- <<PK>> email : char[]
- full_name : char[]
- auth_token : char[]
- password_digest : char[]
- Restaurant has Menu
Ingredient
Ingredient
- portion : double
- Ticket has restaurant
1
- Ticket has Customer
0..*
0..*
0..*
0..*
Ticket
- <<PK>> order_num : int
- order_date : char[]
0..*
Order
1..*
Menu
- <<PK>> item_name : char[]
- price : int
Order
- item_quantity : int

UML -> RM (underline PK, italics FK)

7 Tables
9 FKs

User(email, full_name, auth_token, password_digest)

Restaurant(restaurant_name, location, restaurant_phone, *email*)

Ticket(order_num, *restaurant_name*, order_date, *email*)

Order(*restaurant_name*, *order_num*, *item_name*, item_quantity)

Menu(item_name, *restaurant_name*, price)

Ingredient(*restaurant_name*, *item_name*, *ingredient_name*, portion)

Inventory(ingredient_name, *restaurant_name*, quantity, unit)

BCNF Decomposition

- Attributes: email, full_name, auth_token, password_digest, restaurant_name, location, reataurant_phone, item_name, price, portion, ingredient_name, quantity, unit, order_num, order_date, item_quantity
- A - email
- B - full_name
- C - auth_token
- D - password_digest
- E - restaurant_name
- F - location
- G - restaurant_phone
- H - item_name
- I - price
- J - portion
- K - ingredient_name
- L - quantity
- M - unit
- N - order_num
- O - order_date
- P - item_quantity

Functional Dependencies

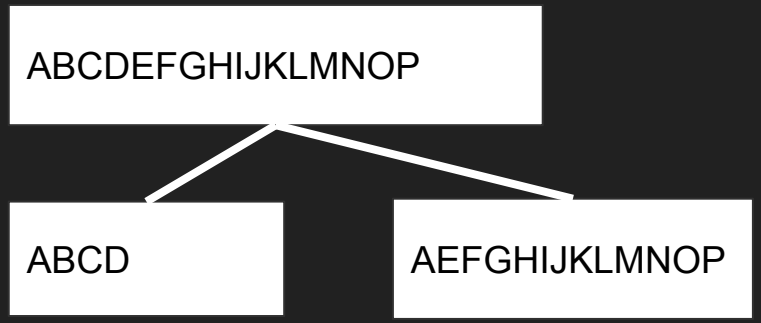
- A -> BCD email -> full_name, auth_token, password_digest
- E -> FG restaurant_name -> location, restaurant_phone
- H -> I item_name -> price
- K - LM ingredient_name -> quantity, unit
- N -> O order_number -> order_date
- EHK -> J restaurant_name, item_name, ingredient_name -> portion
- EHN -> P restaraunt_name, item_name, order_number -> quantity

BCNF Step 1

ABCDEFGHIJKLMNOP
is not in BCNF

A is a superkey of ABCD but not of ABCDEFGHIJKLMNOP therefore

A -> BCD violates BCNF

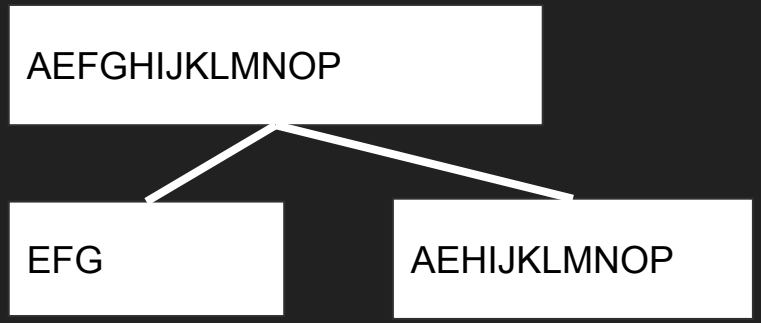


BCNF Step 2

AEFGHIJKLMNOP
is not in BCNF

E is a superkey of EFG but not of AEFGHIJKLMNOP therefore

E -> FG violates BCNF

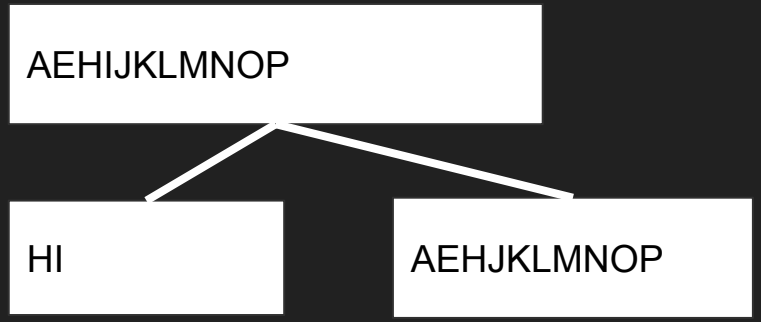


BCNF Step 3

AEHIJKLMNOP
is not in BCNF

H is a superkey of HI but not of AEHIJKLMNOP therefore

H -> I violates BCNF

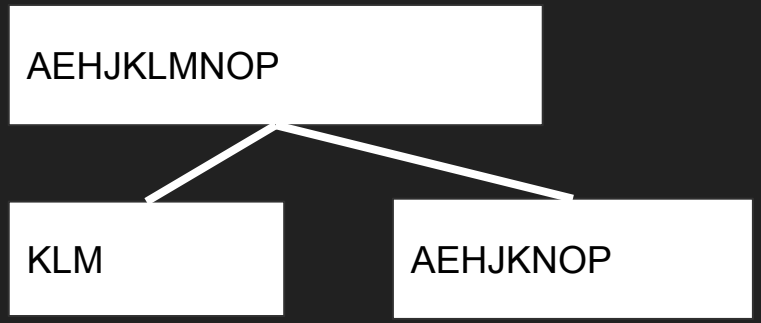


BCNF Step 4

AEHJKLMNOP
is not in BCNF

K is a superkey of KLM but not of AEHJKLMNOP therefore

K -> LM violates BCNF

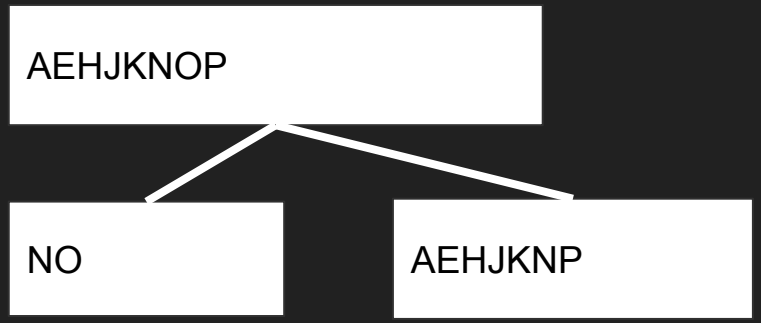


BCNF Step 5

AEHJKNOP
is not in BCNF

N is a superkey of NO but not of AEHJKNOP therefore

N -> O violates BCNF



BCNF Step 6

AEHJKNP
is not in BCNF

EHK is a superkey of EHKJ but not of AEHJKNP therefore

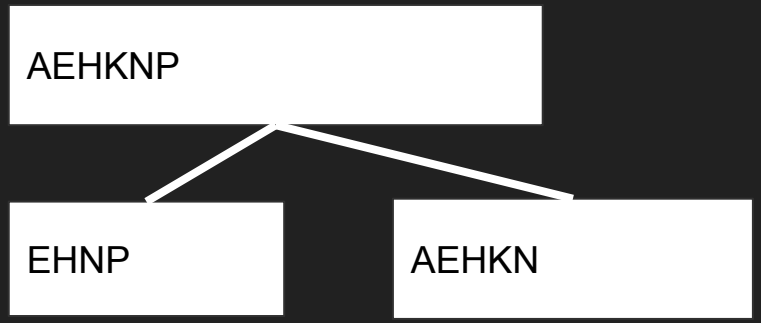
EHK -> J violates BCNF

BCNF Step 7

AEHKNP
is not in BCNF

EHN is a superkey of EHNP but not of AEHKNP therefore

EHN -> P violates BCNF

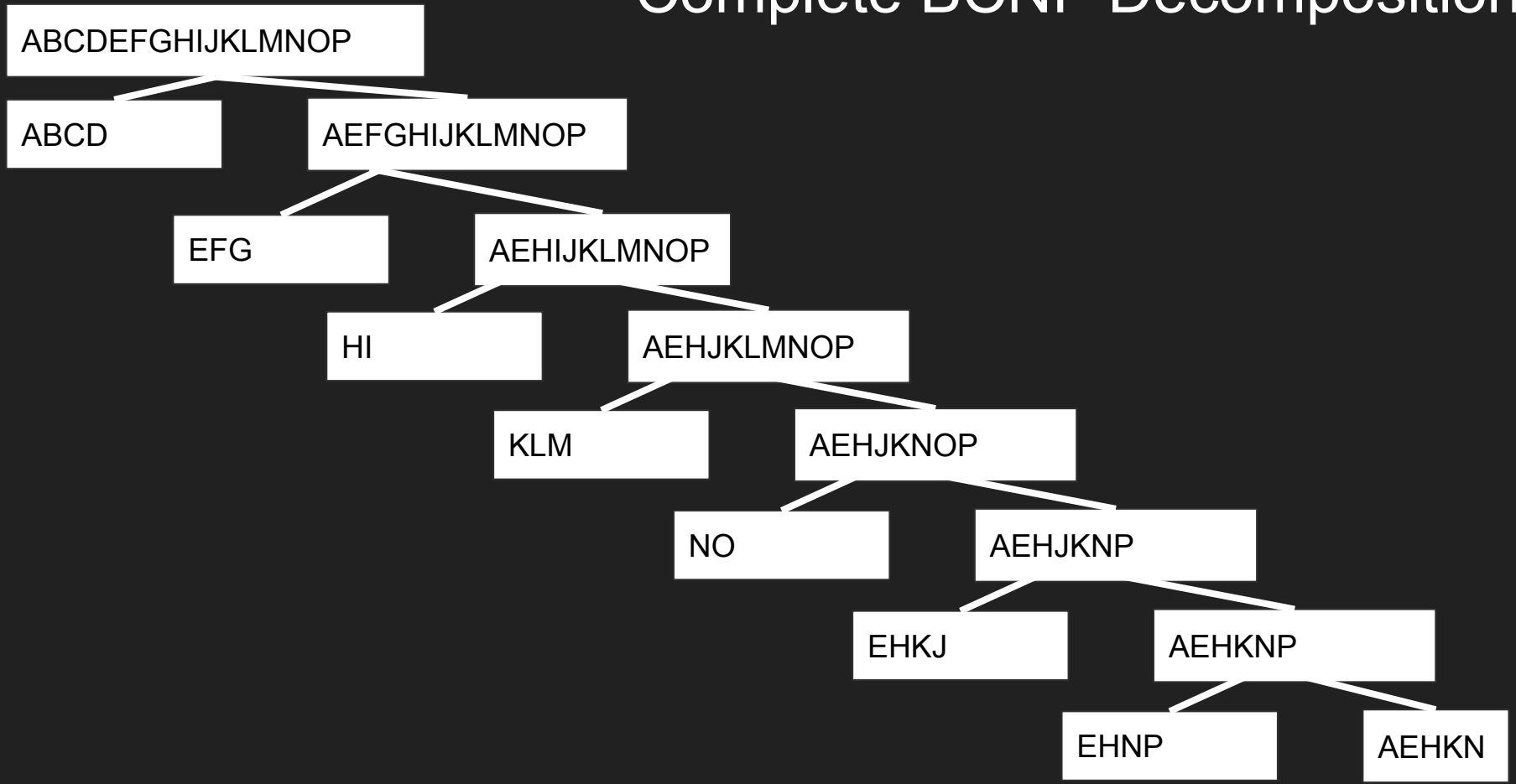


BCNF Final Step

AEHKN is in BCNF

AEHKN is a superkey of AEHKN

Complete BCNF Decomposition



3NF Synthesis

Find Minimal Cover

F = {
- A -> BCD
- E -> FG
- H -> I
- K - LM
- N -> O
- EHK -> J
- EHN -> P

}

Step 1) make all rhs singletons

Fm = {
- A -> B
- A-> C
- A -> D
- E -> F
- E -> G
- H -> I
- K -> L
- K -> M
- N -> O
- EHK -> J
- EHN -> P

}

Step 2) Remove Extraneous Attributes

- None

Step 3) Remove redundant FD’s

- None

3NF Synthesis

Step 1) Merge FD's with same LHS

Fm = {
- A -> B
- A-> C
- A -> D
- E -> F
- E -> G
- H -> I
- K -> L
- K -> M
- N -> O
- EHK -> J
- EHN -> P

}

F = {
- A -> BCD
- E -> FG
- H -> I
- K -> LM
- N -> O
- EHK -> J
- EHN -> P

}

Step 2) Form Tables From the FDs
- R1 = ABCD
- R2 = EFG
- R3 = HI
- R4 = KLM
- R5 = NO
- R6 = EHKJ
- R7 = EHNP

Step 3) Remove Fully Contained Subschemas - None

Step 4) Make sure it's lossless

-No table contains global key

-add table AEHKN

AEHKN+ = ABCDEFGHIJKLMNOP

A(BCD)E(FG)H(I)K(LM)N(O)

EHK ->J EHN ->P

Comparison of RM, BCNF, and 3NF

Both BCNF and 3NF gave us the same tables. They are very similar to our RM schema, but they have the extra superkey table. In practice, we do not need this table, so we chose to the the RM that we generated from our UML.

System Architecture

- Play 2
- Ebeans
- Postgresql
- React

We have a Play 2 server as the backend which uses ebeans to talk to our postgresql database. We wrote the frontend in react which fetches data from the backend using our API.