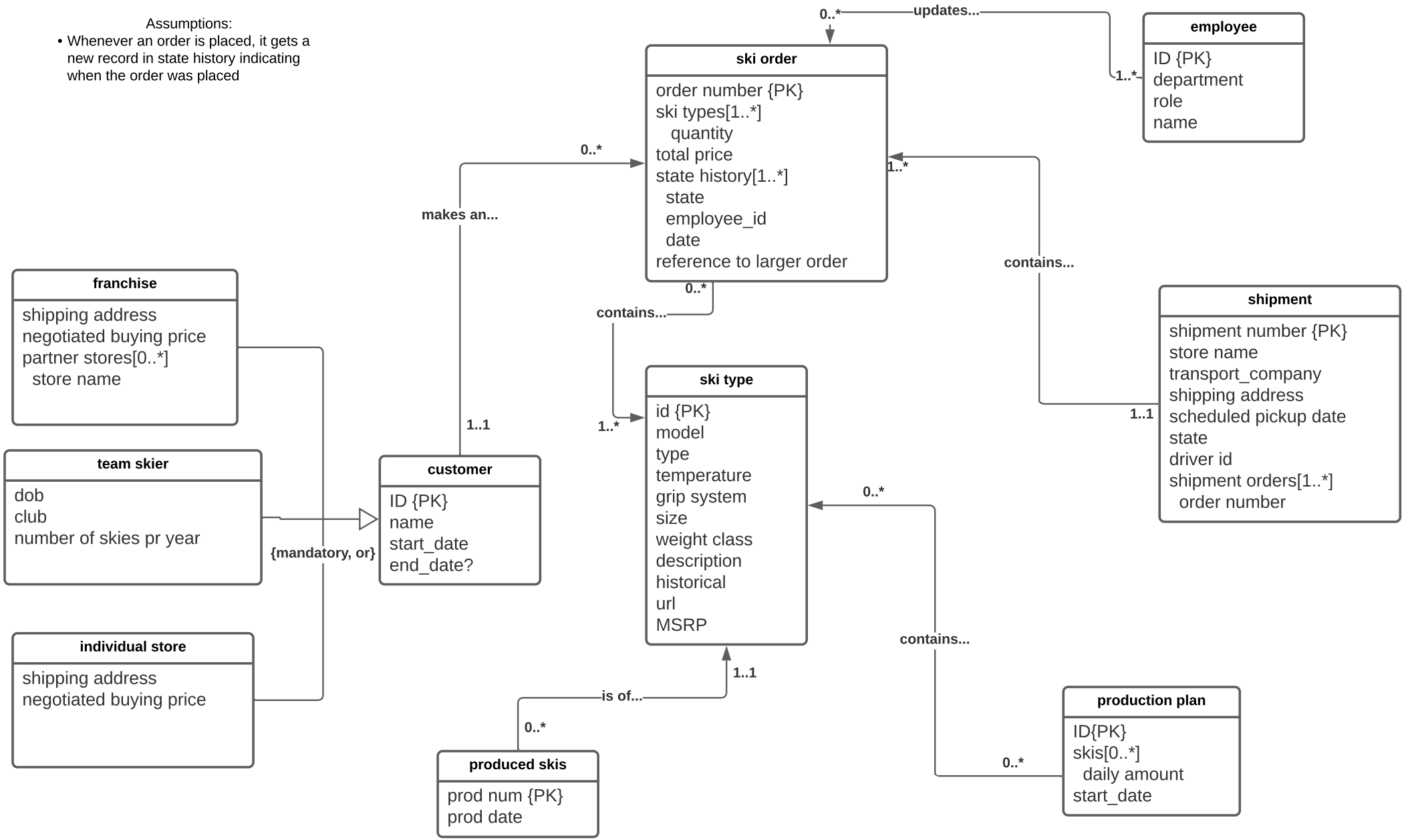

Assumptions:
• Whenever an order is placed, it gets a new record in state history indicating when the order was placed
0..*
updates...
1..*
employee
ID {PK}
department
role
name
ski order
order number {PK}
ski types[1..*]
quantity
total price
state history[1..*]
state
employee_id
date
reference to larger order
0..*
makes an...
1..*
contains...
0..*
contains...
1..1
shipment
shipment number {PK}
store name
transport_company
shipping address
scheduled pickup date
state
driver id
shipment orders[1..*]
order number
franchise
shipping address
negotiated buying price
partner stores[0..*]
store name
1..1
1..*
ski type
id {PK}
model
type
temperature
grip system
size
weight class
description
historical
url
MSRP
team skier
dob
club
number of skies pr year
{mandatory, or}
customer
ID {PK}
name
start_date
end_date?
0..*
individual store
shipping address
negotiated buying price
contains...
1..1
is of...
0..*
produced skis
prod num {PK}
prod date
0..*
production plan
ID{PK}
skis[0..*]
daily amount
start_date

Endpoint definitions

Company endpoints

Customer rep endpoint

Endpoints	Methods	Description
Orders	GET	Retrieving orders based on state. Media type: application/json
Order	PATCH	Change the state of an order. Media type: application/json
Shipment create	POST	Creates a shipment request. Media type: application/json

Endpoints	URI	Description
Orders	http://localhost/rest/com/cus/orders Optional parameter: ?state={state}	Retrieve all orders based on state.
Order	http://localhost/rest/com/cus/orders/{id}	Change the state of an order.
Shipment create	http://localhost/rest/com/cus/req_ship	Request a shipment.

Resource	Representation
Orders	`[order*]`

Order	`{`
	`"id": INTEGER,`
	`"state": STRING`
	`}`

Shipment create	`{`
	`"orders": [order_id*] INTEGER ARRAY`
	`}`

Storekeeper endpoint

Endpoints	Methods	Description
Produced ski	POST	Register a produced ski. Media type: application/json
Get orders	GET	Retrieve orders. Media type: application/json
Update order	PATCH	Update the state of an order. Media type: application/json

Endpoints	URI	Description
Produced ski	http://localhost/rest/com/stor/register	Use this to register a newly produced ski.
Get orders	http://localhost/rest/com/stor/orders Optional parameter: ?state={state}	Use this to get a list of orders.
Update order	http://localhost/rest/com/stor/orders	Use this to update the state of an order.

Resource	Representation

Produced ski	{
	"ski_type": INTEGER,
	"prod_date": STRING
	}

Get orders	[order*]

Update order	{
	"id": INTEGER,
	"state": STRING
	}

Production planner endpoint

Endpoints	Methods	Description
Production plan	POST	Upload production plan. Media type: application/json

Endpoints	URI	Description
Production plan	http://localhost/rest/com/prod/planner	Use this to upload a production plan.

Resource	Representation

Production plan	{
	"start_date": STRING,
	"planned_skis": [
	  "ski_type_id": INTEGER,
	  "daily_amount": INTEGER
	]*
	}

Customer endpoint

Endpoints	Methods	Description
Orders	GET	This resource contains a list of orders they have made. Media type: application/json
Order	GET, DELETE	This resource contains information about a specific order. Media type: application/json
Order create	POST	This resource is for creating orders. Media type: application/json
Split	GET	Request split of order. Media type: application/json
Plan	GET	Contains current four-week production plan. Media type: application/json

Endpoints	URI	Description
Orders	http://localhost/rest/cus/orders Optional query: ?since={date}	Use this to get information about orders.
Order	http://localhost/rest/cus/orders/{id}	Use this to get information about a specific order or update the order.
Split	http://localhost/rest/cus/orders/split/{id}	Use this to request a split of an order. New order ID will be included.
Order create	http://localhost/rest/cus/orders/create	Use this to create a new order.
Plan	http://localhost/rest/cus/plan	Use this to get the current four-week plan.

Resource	Representation
Orders	`[order*]`

Order	`{`
	`"order_number": INTEGER,`
	`"total_price": INTEGER,`
	`"state": STRING,`
	`"reference_to_larger_order": INTEGER,`
	`"customer_id": INTEGER,`
	`"skis": [`
	`  {`
	`  "ski_type_id": INTEGER,`
	`  "quantity": INTEGER`
	`  }`
	`]*`
	`}`

Order create	`{`
	`"total_price": INTEGER,`
	`"state": STRING,`
	`"reference_to_larger_order": INTEGER,`
	`"customer_id": INTEGER,`
	`"skis": [`
	`  "ski_type_id": INTEGER,`
	`  "quantity": INTEGER`
	`]*`
	`}`

Split	`{`
	`"new_id": INTEGER,`
	`"old_id": INTEGER`
	`}`

Plan	`{`
	`"start_date": STRING,`
	`"skis": [`
	`  "ski_type_id": INTEGER,`
	`  "quantity": INTEGER`
	`]*`
	`}`

Transporter endpoint

Endpoints	Methods	Description
Orders	GET	Retrieves information about orders ready for shipment. Change state when picked up. Media type: application/json
Shipments	GET	Retrieves a list of all shipments Media type: application/json
Update shipment	PATCH	Change the state of a shipment. Media type: application/json

Endpoints	URI	Description
Ready orders	http://localhost/rest/trans/orders	Use this to get a list of orders ready for shipment.
Shipments	http://localhost/rest/trans/shipments	Use this to get a list of all shipments
Update shipment	http://localhost/rest/trans/update/{ID}	Use this to change the state of a shipment.

Resource	Representation
Ready orders	`[order*]`

Shipments	`{`
	`"shipment_num": INTEGER,`
	`"store_name": STRING,`
	`"shipping_address": STRING,`
	`"sched_pickup_date": STRING,`
	`"driver_id": INTEGER,`
	`"transport_company": STRING,`
	`"state": STRING,`
	`"orders": [order_id*]: INTEGER ARRAY`
	`}`

Update shipment	`{`
	`"state": STRING`
	`}`

Public endpoint

Endpoints	Methods	Description
Skis	GET	This resource contains a list of the various types of skis. Media type: application/json
Ski	GET	This resource contains information about a specific type of ski. Media type: application/json

Endpoints	URI	Description
Skis	http://localhost/rest/pub/skis Optional query: ?model={name}	Use this to get a list of skis.
Ski	http://localhost/rest/pub/skis/{id}	Use this to get information about a specific ski type.

Resource	Representation
Skis	`[ski*]`

Ski	`{`
	`"id": INTEGER,`
	`"model": STRING,`
	`"temperature": STRING,`
	`"grip_system": STRING,`
	`"size": INTEGER,`
	`"weight_class": STRING,`
	`"description": STRING,`
	`"historical": BOOLEAN,`
	`"url": STRING,`
	`"msrp": INTEGER`
	`}`

1 Logical model

ski_type(id, model, type, temperature, grip_system, size, weight_class, description, historical, url, msrp)
Primary Key id

produced_skis(prod_num, prod_date, ski_type, order_id)
Primary Key prod_num
Foreign Key ski_type *References* ski_type(id)
Foreign Key order_id *References* ski_order(order_number)

production_plan(id, start_date)
Primary Key id

production_plan_ski(id, production_plan_id, ski_type_id, daily_amount)
Primary Key id, production_plan_id, ski_type_id
Foreign Key production_plan_id *References* production_plan(id)
Foreign Key ski_type_id *References* ski_type(id)

employee(id, department, name, role)
Primary Key id

ski_order(order_number, total_price, state, reference_to_larger_order, customer_id)
Primary Key order_number
Foreign Key reference_to_larger_order *References* ski_order(order_number)
Foreign Key customer_id *References* customer(id)

ski_order_ski_type(id, order_id, ski_type_id, quantity)
Primary Key id, order_id, ski_type_id
Foreign Key order_id *References* ski_order(order_number)
Foreign Key ski_type_id *References* ski_type(id)

ski_order_state_history(id, ski_order_id, employee_id, state, date)
Primary Key id, ski_order_id, employee_id
Foreign Key ski_order_id *References* ski_order(order_number)
Foreign Key employee_id *References* employee(id)

shipment(shipment_num, store_name, shipping_address, sched_pickup_date, state, driver_id, transport_company)
Primary Key shipment_num

shipment_orders(id, shipment_num, order_num)
Primary Key id, shipment_num, order_num
Foreign Key shipment_num *References* shipment(shipment_num)
Foreign Key order_num *References* ski_order(order_num)

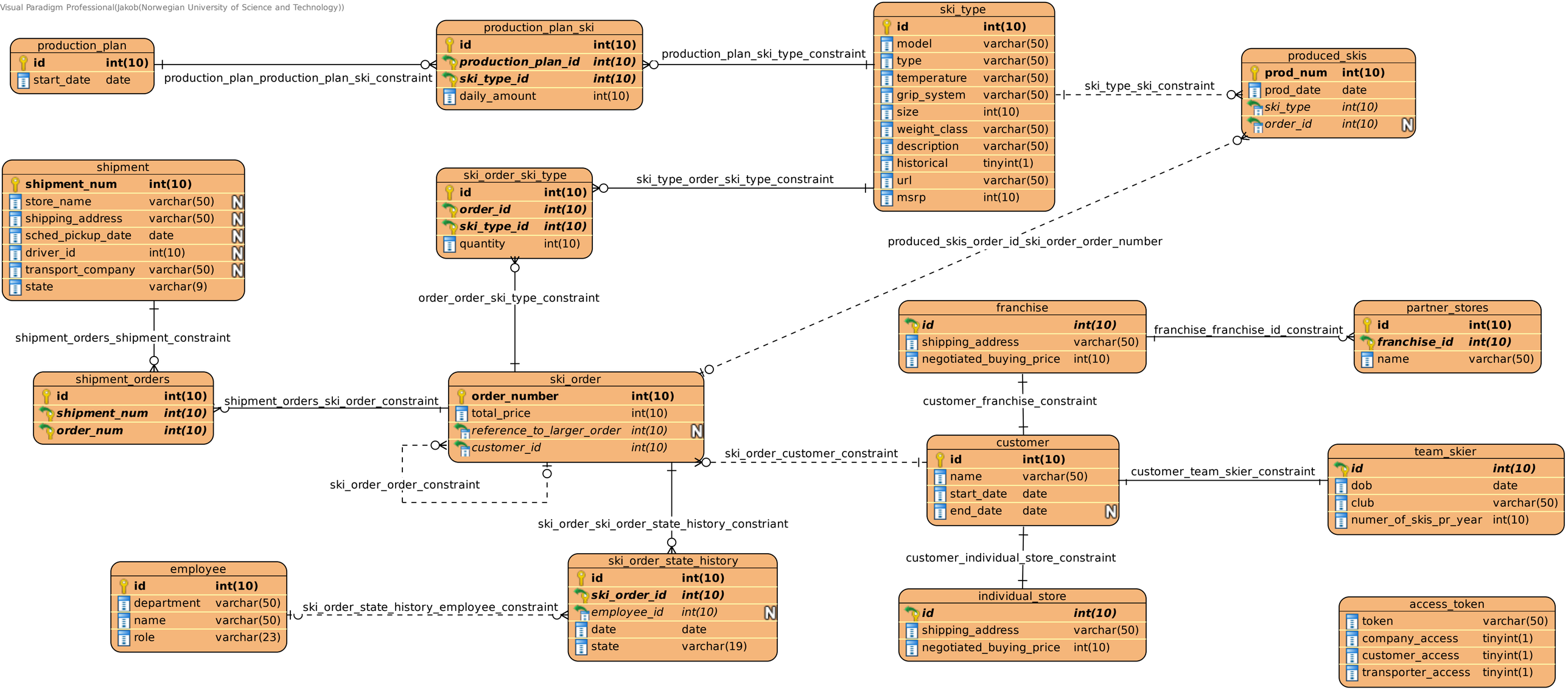
customer(id, name, start_date, end_date)
Primary Key id

individual_store(id, shipping_address, negotiated_buying_price)
Primary Key id
Foreign Key id *References* customer(id)

team_skier(id, dob, club, number_of_skis_pr_year)
Primary Key id
Foreign Key id *References* customer(id)

franchise(id, shipping_address, negotiated_buying_price)
Primary Key id
Foreign Key id *References* customer(id)

partner_stores(id, franchise_id, name)
Primary Key id
Foreign Key franchise_id *References* franchise(id)



NTNU | Norwegian University of Science and Technology

Department of Computer Science

IDATG2204

Database Project

Authors:
Mikkel F. Aas, Jakob F. Karlsmoen, Ruben C. Hegland-Antonsen

March, 2021

1 Test planning

1.1 Definition of test cases

1.1.1 API tests

We want to test every endpoint defined in the endpoint definition document. This includes all the different HTTP methods (GET, PATCH, POST... etc) for each of the endpoints. We will also test that you receive a sensible error response upon errors (for example if an invalid HTTP method is used).

1.1.2 Unit tests

We want to use unit tests during the development to make sure the code we are writing is working as intended. We also want to use unit tests to ensure code quality, for instance in the case of refactoring.

1.2 How

We are planning on using Codeception as a testing framework. It is able to do both unit testing and API testing.

1.2.1 When

Unit tests will be implemented during the development. When merging code with the development branch on GitLab, the tests should be in place.

API tests will be implemented at the end of the development period.

1.2.2 Who

For unit tests, the developer writing the code will be responsible for creating unit tests for that code.

API testing will be divided into three parts and distributed between the developers:

- Company endpoints
- Customer endpoints
- Transporter and public endpoints

At the end we will jointly review the API tests together.