part_warehouse
partNumber (PK)
name
type
cageCode
quantityOrdered
unitPrice

part_warehouse

partNumber	name	type	cageCode	quantityOrdered	unitPrice
10654	Float Control	Plumbing	G413	4	12
10456	Modulator	Electrical	H433	3	7
10776	Hose Assembly	Plumbing	G413	7	9
10657	Float Assembly	Plumbing	G413	5	10

a. attributes: Part Number, Name, Type, Cage Code, Quantity Ordered, Unit Price
assume that the Name is the name of the Part, which is identified by number.

Step 1: Attributes
partNumber, name, type, cageCode, quantityOrdered, unitPrice

Step 2: Definitions
Primary Key (PK): A column or a set of columns that uniquely identifies each row
Candidate Key: The minimal set of attributes that can be potential primary keys. There might be more than one but usually only one is selected as the primary key.
Functional Dependency (FD): An attribute Y is functionally dependent on another attribute X if knowing attribute X determines Y

Step 3: Candidate Keys
In this table, The partNumber attribute uniquely defines each row. Thus, since there are no other candidate keys, this is the primary key, **partNumber**

Step 4: Identify Dependencies
partNumber → name, quantityOrdered, unitPrice, cageCode, type
- The part number determines what is the name of the part, how many of each part is ordered, and what is the price of each part. It also tells wha inventory it is stored at, and the type of part it is

cageCode → type
- The inventory identifier tells what type of part is stored inside

Step 5: What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? Yes. They are all determined by what part number the part is.

So this table is at least in 2NF.
3NF:
- Is the table free of transitive dependencies? (when an attribute depends on another through an indirect relationship) No. The type of part is dependent on the cageCode in the inventory, but both are attributes, so they are dependent by an indrect relationship.

Thus, this table is in 2NF

2NF to 3NF by removing transitive dependencies

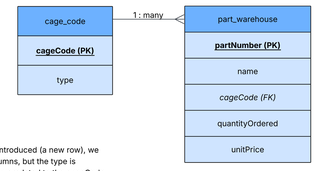
Problem: type is reliant on cageCode while both of these are not primary keys of the part_warehouse entity, creating a transitive dependency.
Solution: Move the dependency cageCode → type to its own table cage_code, and only leave the FK *cageCode* in the part_warehouse entity.

part_warehouse

partNumber	name	*cageCode*	quantityOrdered	unitPrice
10654	Float Control	G413	4	12
10456	Modulator	H433	3	7
10776	Hose Assembly	G413	7	9
10657	Float Assembly	G413	5	10

cage_code

cageCode	type
G413	Plumbing
H433	Electrical
G413	Plumbing
G413	Plumbing



Why 3NF fixes the problem:
In 2NF, for every new part that is introduced (a new row), we would have to add values to 6 columns, but the type is unnecessary to be listed becase it correlated to the cageCode. As long as we know what the cageCode is, we are able to identify what type of part is it. Thus, this additional correlation can be outlined with a new table instead of having to reintroduce the type for every new part. This eliminates redundancy, inconsistency, and anomalies.

appointment
staffNo
therapistName
patNo
patName
appointment date time
branchNo

appointment

staffNo	therapistName	patNo	patName	appointment date time	branchNo
S1011	Fred Smith	P100	Lily White	9/12/2022 10:00	M15
S1011	Fred Smith	P105	Jill Baker	9/12/2022 12:00	M15
S1024	Heidi Pierce	P108	Andy McKee	9/12/2022 10:00	Q10
S1024	Heidi Pierce	P108	Andy McKee	9/14/2022 14:00	Q10
S1032	Richard Levin	P105	Jill Baker	9/14/2022 16:30	M15
S1032	Richard Levin	P110	Jimmy Winter	9/15/2022 18:00	B13

a. attributes: staffNo, therapistName, patNo, patName, appointment data and time, branchNo
assume that there is no variable to put data and time as the same field.

Step 1: Attributes
staffNo, therapistName, patNo, patName, appointment date time, branchNo

Step 2: Definitions
Primary Key (PK): A column or a set of columns that uniquely identifies each row
Candidate Key: The minimal set of attributes that can be potential primary keys. There might be more than one but usually only one is selected as the primary key.
Functional Dependency (FD): An attribute Y is functionally dependent on another attribute X if knowing attribute X determines Y

Step 3: Candidate Keys
In this table, There is no unique value to identify each row. Only the combination of knowing the staffNo, patNo, appointment date, and branchNo, can tell the full information about an appointment to know if the specific staff can handle the patient at the according location and time. Thus, the candidate key = **(staffNo, patNo, appointment date, branchNo)**

Step 4: Identify Dependencies
staffNo →therapistName
- the staff number tells which therapist it is, thus determining the specific therapist name.

patNo → patName
- the patient number tells what the patient's name is

(staffNo, appointment date) → branchNo
- the staff number and appointment date together determines what branch is the therapist at because a therapist can only work at one branch a day

(staffNo, patNo, appointment data and time) → all attributes
- these function together to tell if the therapist can make it to the according date, time, and see the desired patient at the right branch

Step 5: What normal form is it in
1NF:
- Atomic values? No. Appointment date and time should not be in one column. This makes 2 separate types of variables in 1 column, which is wrong.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

Thus, this table is not in a Normal Form.

0NF to 1NF by allowing only atomic values

Problem: appointment date and time are in the same column even though they are two separate variable types. This creates a problem for not only the entity unable to be converted to 1NF, but also because each patient is allowed to request an appointment with the therapist only on the dates that they are working at a particular branch, and therapists only work at one branch a day. However, this is unaffiliated with appointment time, which is why we need to separate these two variables.
Solution: we can make it in 1NF by separating this component into appointmentDate and appointmentTime

appointment

staffNo	therapistName	patNo	patName	appointmentDate	appointmentTime	branchNo
S1011	Fred Smith	P100	Lily White	9/12/2022	10:00	M15
S1011	Fred Smith	P105	Jill Baker	9/12/2022	12:00	M15
S1024	Heidi Pierce	P108	Andy McKee	9/12/2022	10:00	Q10
S1024	Heidi Pierce	P108	Andy McKee	9/14/2022	14:00	Q10
S1032	Richard Levin	P105	Jill Baker	9/14/2022	16:30	M15
S1032	Richard Levin	P110	Jimmy Winter	9/15/2022	18:00	B13

now, the primary key can change into the correct form, which is, **(staffNo, patNo, appointmentDate, branchNo)**

What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? No. The primary key is composite, but therapistName only depends on staffNo, patName only depends on patNo, and appointmentTime and branch number only depends on appointmentDate. These are all partial dependencies that need to be solved before we can say it is in 2NF.

This table is in 1NF.

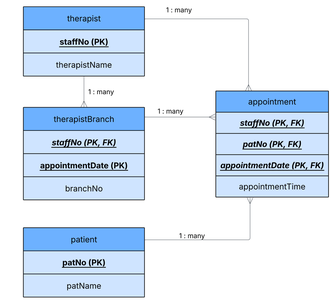
appointment
staffNo
therapistName
patNo
patName
appointmentDate
appointmentTime
branchNo

1NF to 2NF by removing partial dependencies

Problem: therapistName is only dependent on the staff number, patient name is only dependent on the patient number, and branch number is only dependent on the combination of which therapist is working on what date. These need to be solved before the table can be in 2NF.
Solution: Convert the appointment table into 4 tables including itself, a table for therapist to correlate staff number to the therapist's name, a table for patient to correlate patient number with the patient name, and a table for therapistBranch to identify what branch the therapist is working at on the date of. We can put all of the neccessary information from therapist, patient, and therapistBranch identifiers into the appointment table in order to find the details from these smaller tables. The appointment will have ***staffNo patNo***, and ***appointmentDate*** as foreign keys which also make up its primary key to uniquely represent each appointment. The therapist table will use **staffNo** as its PK, patient table will use **patNo**, and therapistBranch will use **(*staffNo*, appointmentDate)** with ***staffNo*** being a FK from therapist in order to uniquely represent its rows.

appointment

staffNo	*patNo*	*appointmentDate*	appointmentTime
S1011	P100	9/12/2022	10:00
S1011	P105	9/12/2022	12:00
S1024	P108	9/12/2022	10:00
S1024	P108	9/14/2022	14:00
S1032	P105	9/14/2022	16:30
S1032	P110	9/15/2022	18:00



What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? Yes. There are no attributes that is particially determined by the primary key.

So this table is at least in 2NF.
3NF:
- Are the tables free of transitive dependencies? (when an attribute depends on another through an indirect relationship) Yes. Since each table has at most 1 attribute that is not the primary key, it is impossible for a non-primary attribute to depend on another non-primary attribute. Furthermore, as we have already separately the tables so they are only grouped based on what is related to their respected primary keys, there are no transitive dependencies.

These tables are in 3NF

therapist

staffNo	therapistName
S1011	Fred Smith
S1011	Fred Smith
S1024	Heidi Pierce
S1024	Heidi Pierce
S1032	Richard Levin
S1032	Richard Levin

patient

patNo	patName
P100	Lily White
P105	Jill Baker
P108	Andy McKee
P108	Andy McKee
P105	Jill Baker
P110	Jimmy Winter

therapistBranch

staffNo	appointmentDate	branchNo
S1011	9/12/2022	M15
S1011	9/12/2022	M15
S1024	9/12/2022	Q10
S1024	9/14/2022	Q10
S1032	9/14/2022	M15
S1032	9/15/2022	B13

contract
eNo
contractNo
hours
eName
eventNo
eventLoc

contract_hours

eNo	contractNo	hours	eName	eventNo	eventLoc
1135	C1024	16	Smith J	H25	Queens
1057	C1024	24	Hocine D	H25	Queens
1068	C1025	28	White T	H4	Yonkers
1135	C1025	15	Smith J	H4	Yonkers
1135	C1026	10	Smith J	H25	Queens

Step 1: Attributes: eNo, contractNo, hours, eName, eventNo, eventLoc

Step 2: Definitions
Primary Key (PK): A column or a set of columns that uniquely identifies each row
Candidate Key: The minimal set of attributes that can be potential primary keys. There might be more than one but usually only one is selected as the primary key.
Functional Dependency (FD): An attribute Y is functionally dependent on another attribute X if knowing attribute X determines Y

Step 3: Candidate Keys
In this table, There is no unique value to identify each row. Only the combination of knowing the employee number and contract number can help identify the specific contract and number of hours accumulated for the specific employee. Thus, the candidate key should be a composite key **(eNo, contractNo)**

Step 4: Identify Dependencies
eNo →eName
- the employee number is unique for each member of staff

eventNo → eventLoc
- the event number can identify event location, every event number will be assign to one event location

(eNo, contractNo) → hours
- the combination of employee number and contract number can identify the number of hours worked, since any of these alone cannot specific the particular event they worked on or the member linked with it.

contractNo → eventNo
- each contract only applies to one event

Step 5: What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? No. The employee name is only reliant on the employee number, which is only a part of the composite key. The event number is reliant on the contractNo, which is also only a part of the composite key.

So this table is in 1NF.

Problem: eName is only dependent on **eNo** and eventNo is only reliant on the **contractNo**, while **eNo** and **contractNo** make up a composite PK.

Solution: split contract table into 3 tables. One for contract, one for employee, and one for contract_hours, where eName and eventNo can be made attributes under event and contract respectively to shape the diagram into 2NF. **eNo** can be the primary key of both contract_hours and event and become a foreign key under contract_hours, and **contractNo** can be a primary key under contract_hours and contract. Contract hours can be shortened to not include the dependencies on partial composite keys, and its composite primary key **(eNo, contractNo)** will be FKs from employee and contract table

contract_hours

eNo	contractNo	hours	eventLoc
1135	C1024	16	Queens
1057	C1024	24	Queens
1068	C1025	28	Yonkers
1135	C1025	15	Yonkers
1135	C1026	10	Queens

employee

eNo	eName
1135	Smith J
1057	Hocine D
1068	White T
1135	Smith J
1135	Smith J

contract

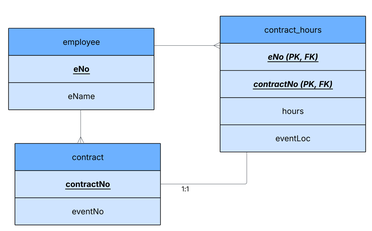
contractNo	eventNo
C1024	H25
C1024	H25
C1025	H4
C1025	H4
C1026	H25

What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? Yes. They are all determined by what part number the part is.

So this table is at least in 2NF.
3NF:
- Is the table free of transitive dependencies? (when an attribute depends on another through an indirect relationship) No. The event location is still an attribute under contract_hours when it is only dependent on the event number, which was previously a non-primary attribute value of contract_hours now under contract. Thus, this is a transitive dependency.

Thus, this table is in 2NF



Problem: eventLoc has a transitive dependency on eventNo, which was previous a non-primary attribute of contract_hours, but now an attribute of contract.

Solution: move the attribute eventLoc to be an attribute of a new entity event, which has the **eventNo** as the PK, since that is what eventLoc depends on. We cannot put it under the same entity as the current eventNo, contract, because the eventLoc is unrelated to the **contractNo**, which is the PK of the entity. Then, change the **eventNo** which is currently an attribute of contract to be a FK from event in order to reorganize this diagram into 3NF.

contract_hours

eNo	contractNo	hours
1135	C1024	16
1057	C1024	24
1068	C1025	28
1135	C1025	15
1135	C1026	10

event

eventNo	eventLoc
H25	Queens
H25	Queens
H4	Yonkers
H4	Yonkers
H25	Queens

employee

eNo	eName
1135	Smith J
1057	Hocine D
1068	White T
1135	Smith J
1135	Smith J

contract

contractNo	*eventNo*
C1024	H25
C1024	H25
C1025	H4
C1025	H4
C1026	H25

What normal form is it in
1NF:
- Atomic values? Yes. Each column only has one value per row.
- No repeating groups of rows and columns? Yes. There are no columns or rows where there is the same meaning of data stored.

So this table is at least in 1NF.
2NF:
- Is every non-primary attribute fully dependent on the primary key? Yes. They are all determined by what part number the part is.

So this table is at least in 2NF.
3NF:
- Is the table free of transitive dependencies? (when an attribute depends on another through an indirect relationship) Yes. There are no non-primary attributes dependent on other non-primary attributes.

Thus, this table is in 3NF