Engineering Databases

Lecture 7 – Normalization 1, 2 & 3

January 11, 2023

M. Saeed Mafipour & Mansour Mehranfar

Content of lecture 6

- Pitfalls in SQL
  - Use ` and ‘ and “ and \
  - Be aware of attacks on Databases, e.g. SQL Injections
- Sort result of a query using ORDER BY
- Limit the number of rows of sorted query by LIMIT

- Triggers
  - Mechanism to react on updates, insert, and delete statements
  - Is connected to changes of the content of a table
  - Will run before or after these events
  - Runs a single or multiple statements
  - Will run for each row. This means for all rows that activate the trigger

Content of Lecture 6

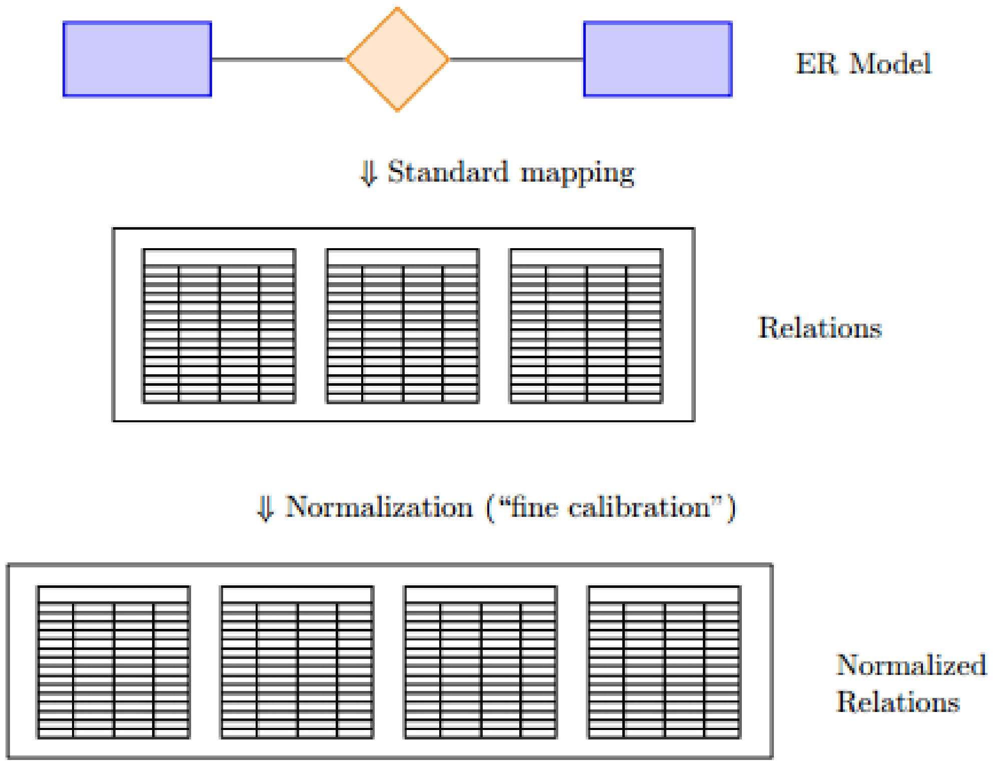
- Views are virtual tables
- Views ‚save‘ SQL select statements

- Transaction ensure database consistency
- Transaction bundle multiple operations in a single unit
- ACID = Atomicity, Consistency, Isolation, and Durability

Normalization

- Motivation

- Create a better schema
  by **means of formal methods**
- avoid anomalies when
  updating/inserting/deleting tuples
- avoid redundancy



- Nice source:
  http://www.bkent.net/Doc/simple5.htm

Normalization

Light orange is a
foreign key reference

Orange is the
primary key

- Why do we do that?

Album	Musician	Date	Track Title	Track number

CD #	Album	Date
1	Album Title	1999
2	Album Title	1999
3	Album Title	1999

CD #	Musician #
1	31
2	32
3	32

Musician #	Musician
31	Musician 1
32	Musician 2

CD #	Tack #
1	101
1	102
3	103

Track #	Track Title	Track number
101	Track Title	1
102	Track Title	2
103	Track Title	1

Normalization

- First Normal Form (1NF)
- All attributes must be atomic

Not 1NF:

CD #	Album Title	Musician	Date	Tracks
1	Album Title	Musician	1999	1 Track Title, 2 Track Title
2	Album Title	Musician	1999	1 Track Title
3	Album Title	Musician	1999	1 Track Title

The tracks are not atomic

1NF:

CD #	Album Title	Musician	Date	Track#	Tracks
1	Album Title	Musician	1999	1	Track Title
1	Album Title	Musician	1999	2	Track Title
2	Album Title	Musician	1999	1	Track Title
3	Album Title	Musician	1999	1	Track Title

Normalization – NF1 (example)

- Change the track title ‘Run’ to ‘Runner’ for CD#1

CD #	Album Title	Musician	Date	Tracks
1	Magic	A	1999	Trick, Unity, Run, Runner
2	Dragon	B	1999	Fire, Smoke, Gold
3	Dance	A	1999	Neon, Light, Floor

- The system finds the text: **Trick, Unity, Run, Runner**
- It has to scan for Run and exchanges Run with Runner
- The result is: **Trick, Unity, Runner, Runnerner**

- Normal Form 1: All attributes must be atomic
- Solutions: Never put distinct data items in an single attribute

Normalization

- Second Normal Form (2NF)
- All non-key attributes are fully functional dependent on all key attributes

Not 2NF:

CD #	Album	Musician	Date	Track#	Title
1	Album Title 1	Musician 1	1999	1	Tack Title
1	Album Title 1	Musician 1	1999	2	Tack Title
2	Album Title 2	Musician 2	1999	1	Tack Title
3	Album Title 3	Musician 3	1999	1	Tack Title

2NF:

CD #	Album	Musician	Date
1	Album Title	Musician	1999
2	Album Title	Musician	1999
3	Album Title	Musician	1999

CD #	Track#	Title
1	1	Tack Title
1	2	Tack Title
2	1	Tack Title
3	1	Tack Title

Normalization – NF2 (example)

- Change the artist ‘A’ hometown to ‘Washington’

Genre	Artist	Hometown
Dance	A	New York
Rock	B	Berlin
Pop	A	New York

→

Artist	Hometown
A	New York
B	Berlin

Genre	Artist
Dance	A
Rock	B
Pop	A

- The system have to find all rows that correspond to artist A
- It has to scan the whole table and rename the Hometown multiple times for A.

- Normal Form 2: All non-key attributes are **dependent on** the complete **primary key**
- Here: Hometown (non-key) is dependent on Artist (key) but not on Genre (key)
- Solution: Split into separate tables in which this is true

Normalization – NF3

- The 3 normal form provides optimal balance between performance, redundancy and flexibility

Artist	Birth	Zipcode	Hometown
A	1975	503	New York
B	2003	313	Berlin
C	1993	313	Berlin

$\Rightarrow$

Artist	Birth	Zipcode
A	1975	503
B	2003	313
C	1993	313

Zipcode	Hometown
503	New York
313	Berlin

- Normal Form 3: No non-key attribute is **transitively dependent on** a **non-key attribute**
- Here:
  - Zipcode (non-key) is dependent on Artist (key). (Z depends on A)
  - Birth (non-key) is dependent on Artist (key). (B depends on A)
  - Hometown (non-key) is dependent on Artist (key). (H depends on A)
  - However, Hometown is actually dependent on Zipcode (H depends on Z)
- Solution: Break (H depends A, via Z depends A) by splitting the table

Normalization - BCNF

- Boyce-Codd-Normalform (BCNF) is an extension of the 3NF

Artist	Album Title	Album Style
A	1X	Rock
A	2X	Dance
B	1Y	Rock

Artist	Album Title
A	1X
A	2X
B	1Y

Album Title	Album Style
1X	Rock
2X	Dance
1Y	Rock

- BCNF is given if such ambiguous primary keys are removed
- BCNF: All redundancies based on dependencies has been removed
- Here:

  primary key candidates are (Artist, Album Title) and (Album Title, Album Style)

Normalization

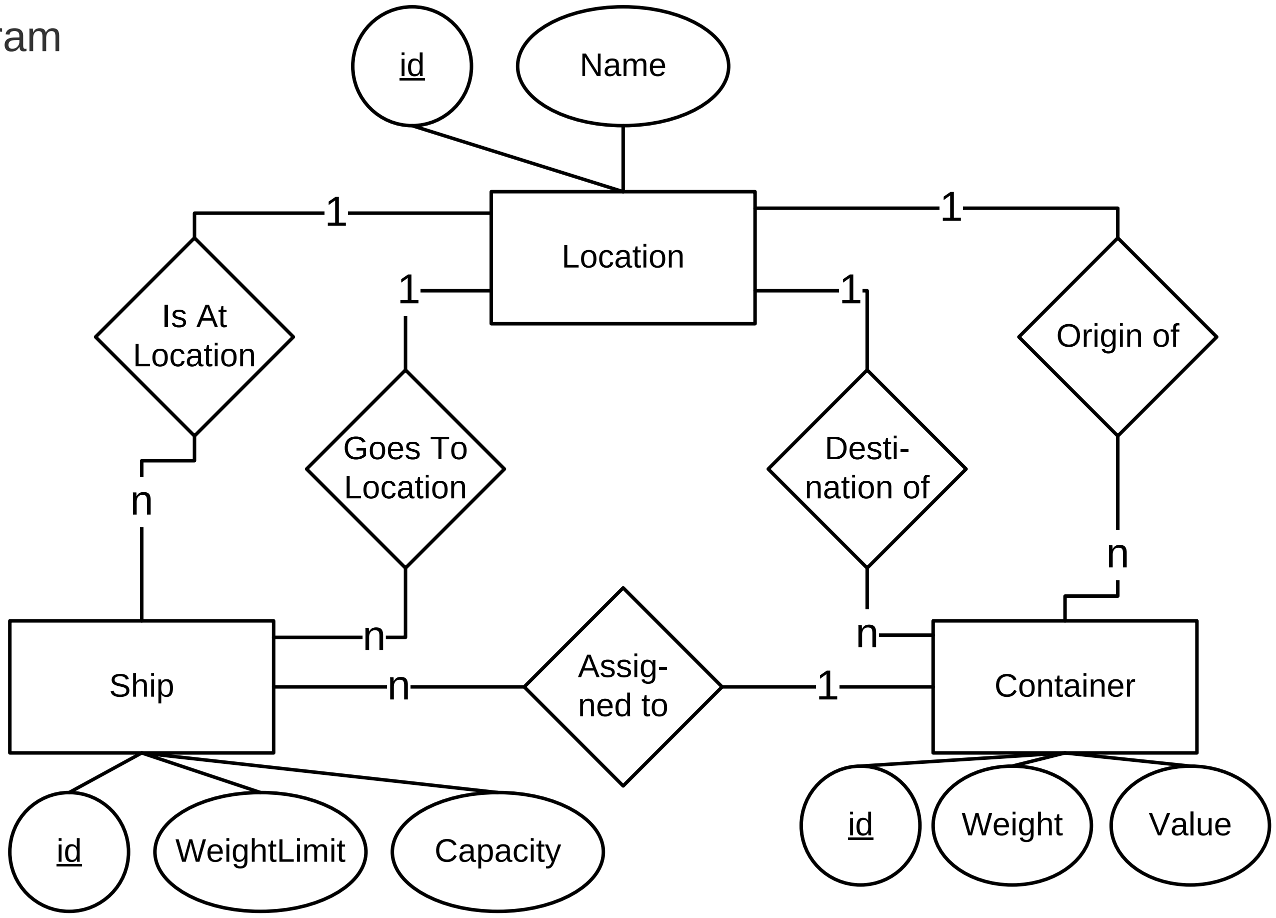
- How to reach 3NF

- Create a ER diagram
  - Identify entities and relation
  - Find attributes and primary keys for your entities
  - Artificial primary keys (ids) are often beneficial
  - Use atomic attributes (no list of items in a single attribute)
  - Make sure that all attributes of an entity are dependent on the primary key
  - Thus, never create a dependencies between unrelated attributes
  - If you have multiple primary key candidates, question your design

- The result will be a 3NF table (mostly also in BCNF)

Exercise – Freight Company Discussion

- ER-Diagram



Exercise – Freight Company Discussion

- Tables without N:1 reduction

shipsAtLocation	
shipId	locationId

shipGoingToDestination	
shipId	locationId

locations	
locationId	locationName

ships		
shipId	weightCapacity	containerCapacity

containers				
containerId	weight	value	origin	destination

containerOnShip	
shipId	containerId

Exercise – Freight Company Discussion

- Tables including N:1 reduction

ships				
shipsId	weightCapacity	containerCapacity	origin	destination

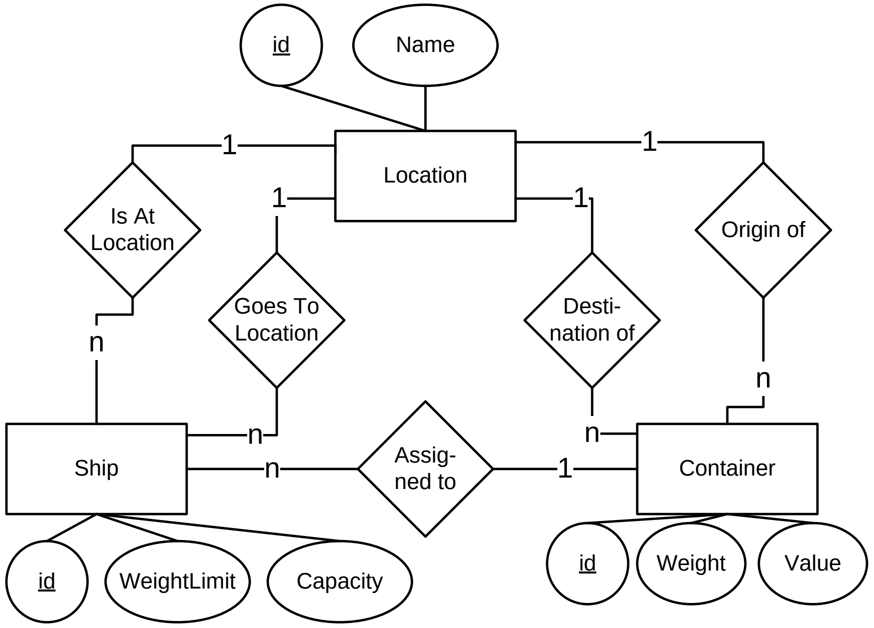
locations	
locationId	locationName

containers					
containersId	weight	value	origin	destination	shipId

Exercise – Freight Company Discussion (no 1:N reduction)

- List the locationId for each ship

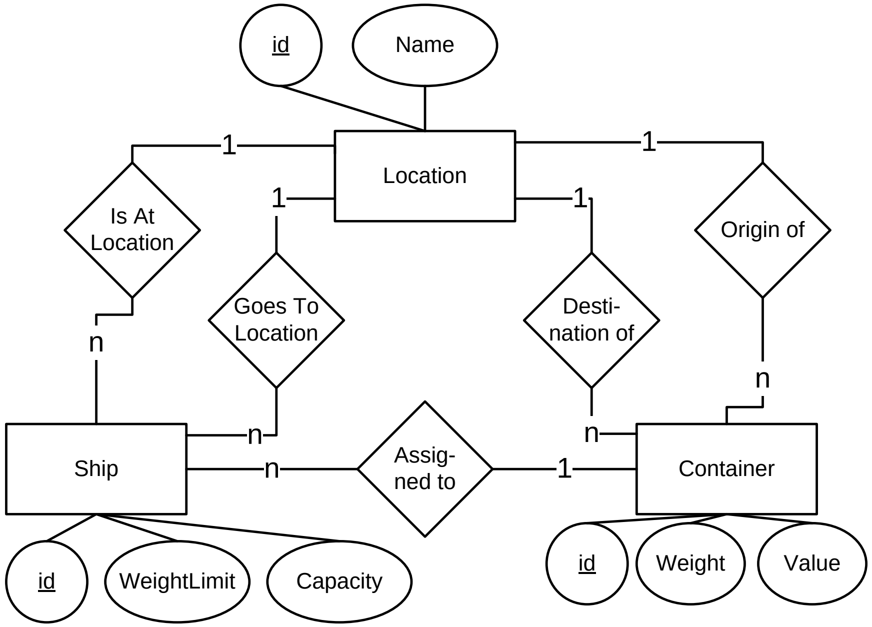
```
SELECT * FROM ships NATURAL JOIN shipAtLocation;
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List the locationName for each ship

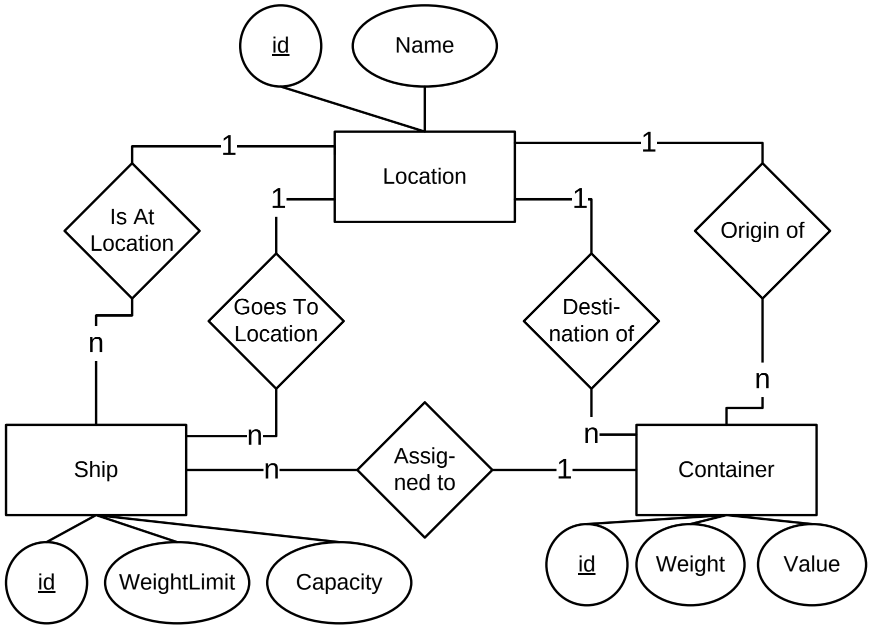
```
SELECT * FROM ships
NATURAL JOIN shipAtLocation
JOIN locationsON shipAtLocation.origin_id=locations.locationId;
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List the shipId in Hamburg

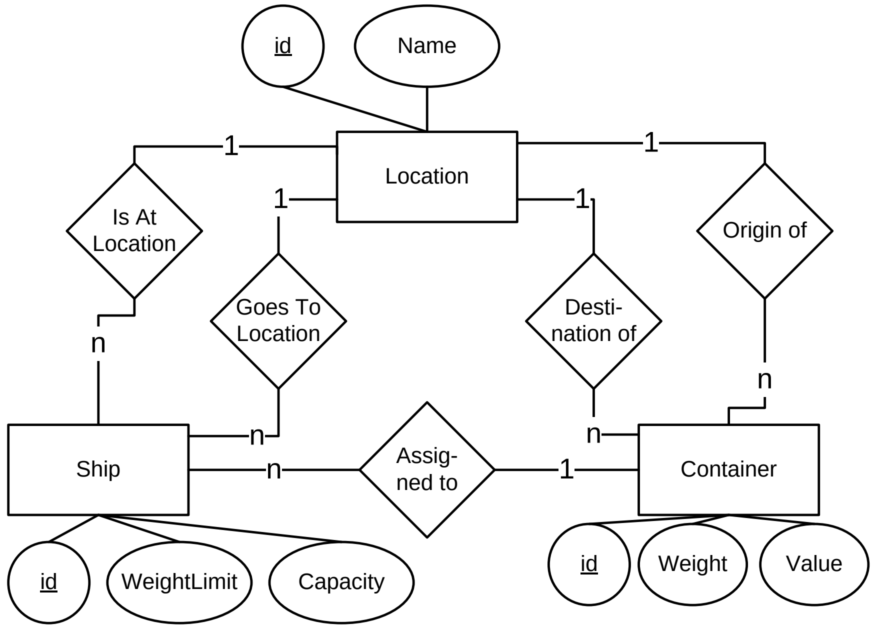
```
SELECT * FROM ships NATURAL JOIN shipAtLocation
JOIN locations
ON shipAtLocation.originId=locations.locationId
WHERE locationName='Hamburg';
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List all containers on ship 1

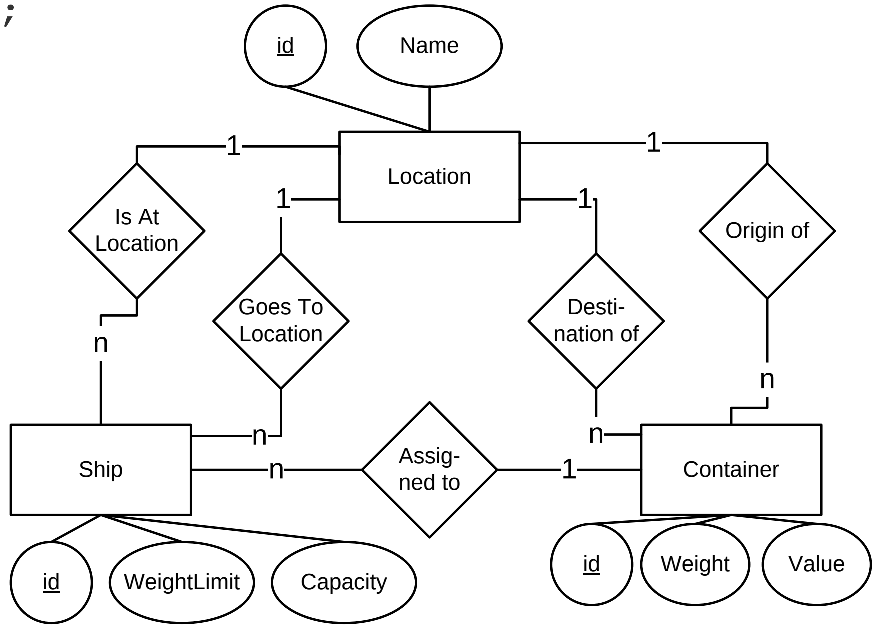
```
SELECT * FROM containerOnShip WHERE shipId=1
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List all containers on ships in Hamburg

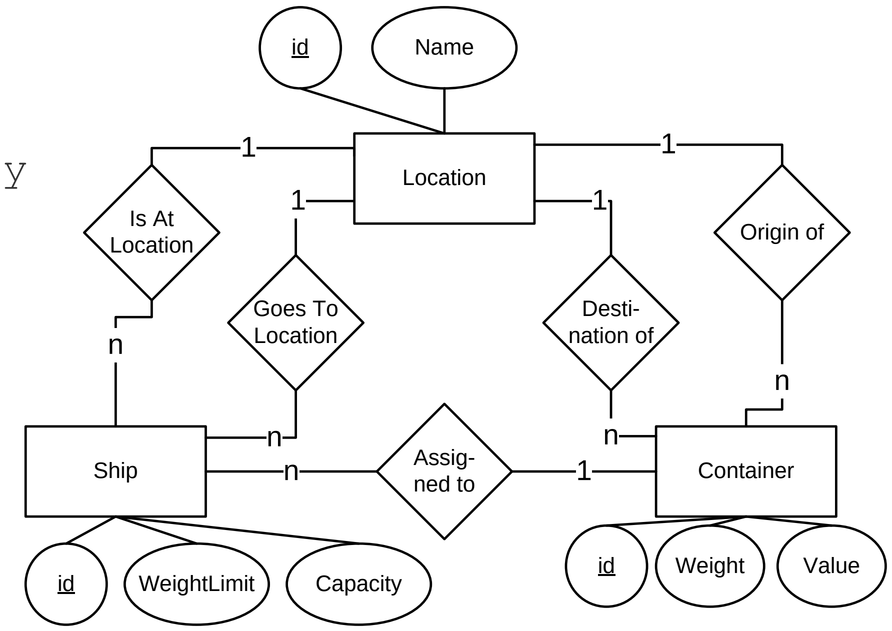
```
SELECT * FROM containerOnShip
NATURAL JOIN shipAtLocation
JOIN locations ON shipAtLocation.originId=locations.locationId
WHERE locationName='Hamburg';
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List all ships which have reached their weight capacity
  (sum of container weights on ship are equal to the weight capacity)

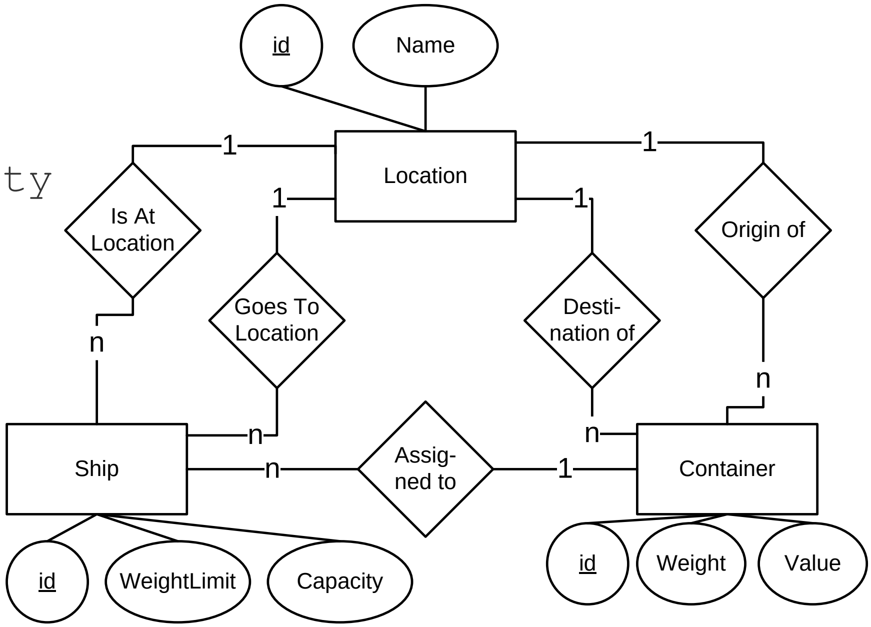
```
SELECT * FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
GROUP BY shipId
HAVING SUM(weight)=weightCapacity
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List all ships which have reached their capacity by either weight or container count

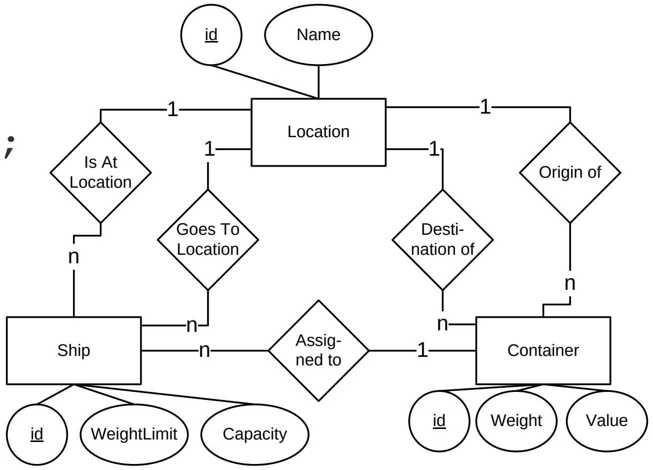
```
SELECT * FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
GROUP BY shipId
HAVING SUM(weight)=weightCapacity
OR
COUNT(containerId)=containerCapacity
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List all ships which have reached their capacity by either weight or container count
  Extension 1: Also show the location of the ships

```
SELECT * FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
NATURAL JOIN shipAtLocations
JOIN Location ON shipAtLocation.originId=locations.locationId
GROUP BY shipId
HAVING SUM(weight)=weightCapacity
OR COUNT(containerId)=containerCapacity;
```

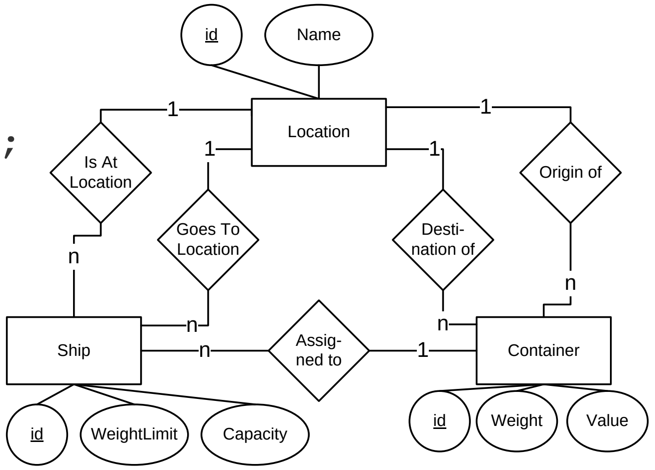


Exercise – Freight Company Discussion (no 1:N reduction)

- List all ships which have reached their capacity by either weight or container count

  Extension 2: Also show the total weight of each ship as a column

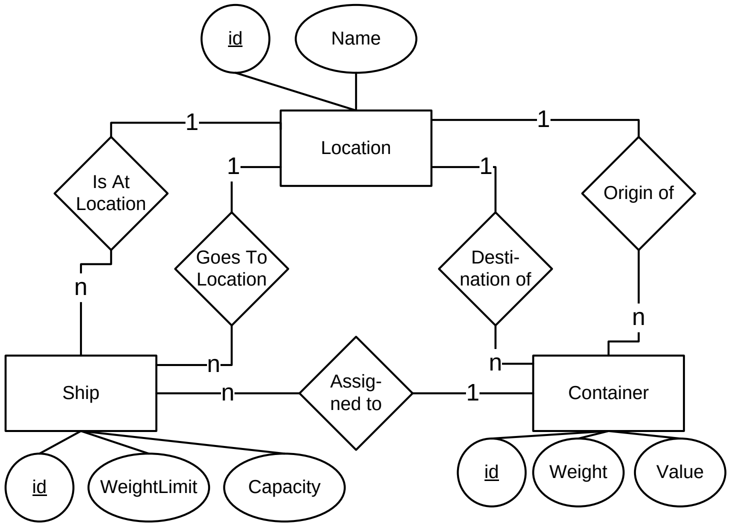
```
SELECT *,SUM(weight) FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
NATURAL JOIN shipAtLocations
JOIN Location ON shipAtLocation.originId=locations.locationId
GROUP BY shipId
HAVING SUM(weight)=weightCapacity
OR COUNT(containerId)=containerCapacity;
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List the lightest ship in Hamburg

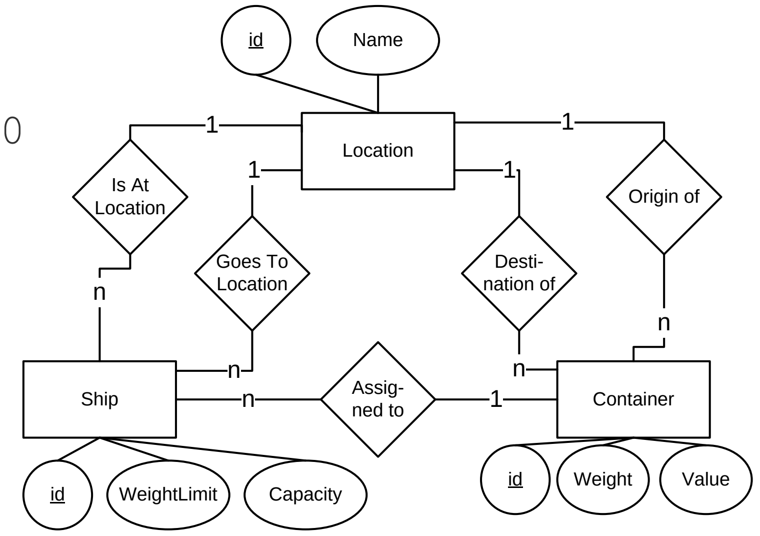
```
SELECT *, SUM(weight) FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
NATURAL JOIN shipAtLocation
JOIN Location ON shipAtLocation.originId=locations.locationId
WHERE locationName='Hamburg'
GROUP BY shipId
ORDER BY SUM(weight)
LIMIT 1
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List the lightest ship in Hamburg which has a free weight capacity of 100

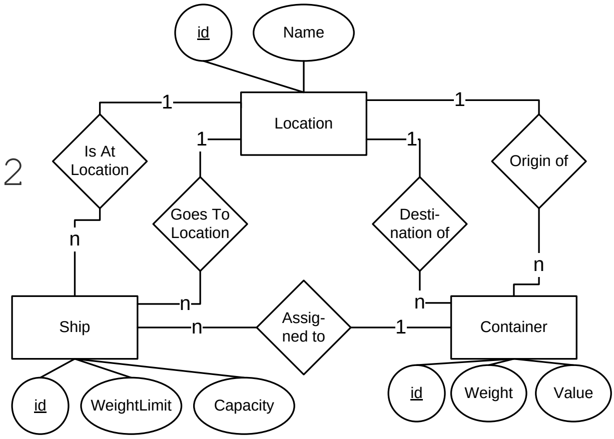
```
SELECT *,SUM(weight) FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
NATURAL JOIN shipAtLocation
JOIN locations ON shipAtLocation.originId=locations.locationId
WHERE locationName='Hamburg'
GROUP BY shipId
HAVING weightcapacity – SUM(weight)>100
ORDER BY SUM(weight)
LIMIT 1
```



Exercise – Freight Company Discussion (no 1:N reduction)

- List the lightest ship in Hamburg which has a free weight capacity of 100 and a free container capacity of 2

```
SELECT *,SUM(weight) FROM containerOnShip
NATURAL JOIN ships
NATURAL JOIN containers
NATURAL JOIN shipAtLocation
JOIN locations ON shipAtLocation.originId=locations.locationId
WHERE locationName='Hamburg'
GROUP BY shipId
HAVING weightCapacity - SUM(weight)>=100
AND containerCapacity - COUNT(containerId)>=2
ORDER BY SUM(weight)
LIMIT 1
```



Exercise – Freight Company – More

- List all ships and their respective origin and destination
- Add a new ship with weight capacity 100, container capacity 10, that is currently in Hamburg
- List all ships that do not have a destination assigned.

- Create a trigger that assigns a newly added container to the lightest ship matching origin, destination and not exceeding the ship's capacity.

End of Lecture

Thank you for your attention