Tampere University

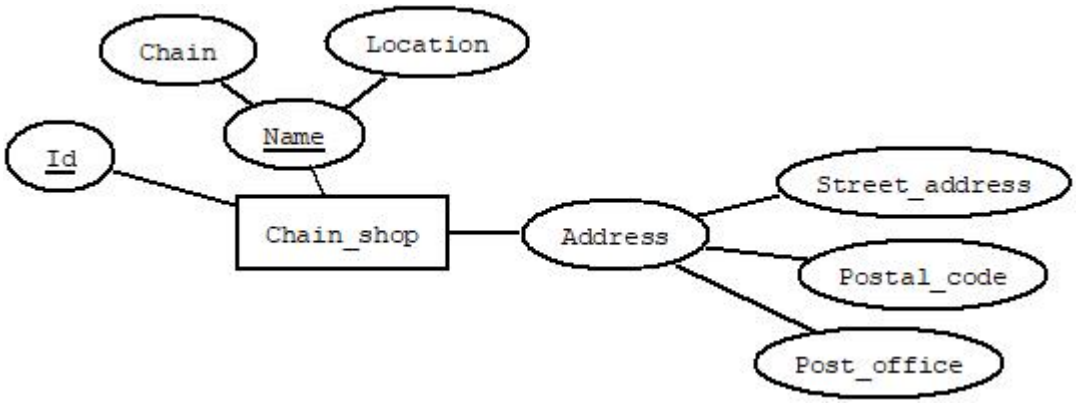
Database basics

ER modelling - continued

Composite attribute as key
N-ary relationship types
Keys and primary keys
Design principles

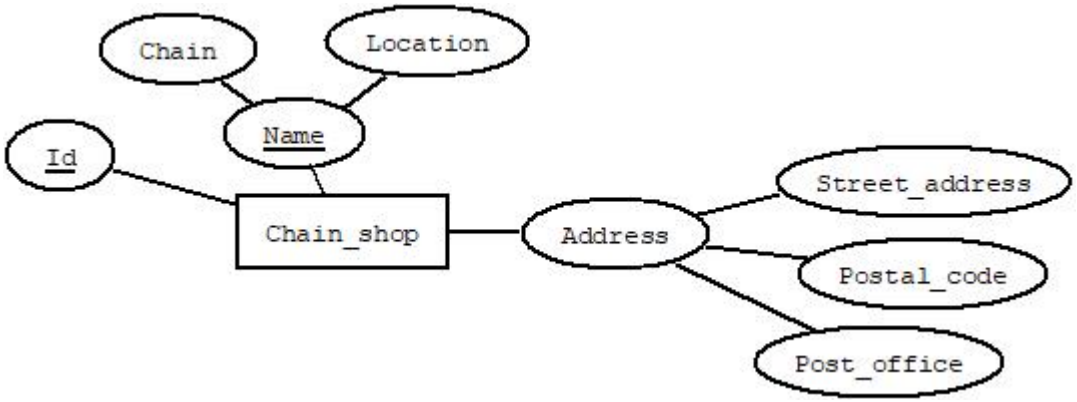
Composite attribute as key (1/3)

- It was previously stated that a composite attribute consists of two or more simple attributes and can be referred to either as a whole or as parts.
- In the example below, the chain shop (chain store) entity type has two composite attributes:
  - Name, which is one of the keys of the entity type
    - Chain
    - Location
  - Address
    - Street_address
    - Postal_code
    - Post_office



Composite attribute as key (2/3)

- Composite attributes of the chain shop entity type are mapped to table columns by adding a separate column for each simple attribute in the composite attribute.
- Of the keys of the chain shop entity type, ID is selected as the primary key, and the chain and location combination is made a key (unique).
- If there were multiple keys in the table, one could use annotation U#, where # is a number, to indicate the columns of different keys.



chain_shop
°id PK
°chain U
°location U
°street_address
°postal_code
°post_office

chain_shop
°id PK
°chain U1
°location U1
°street_address
°postal_code
°post_office

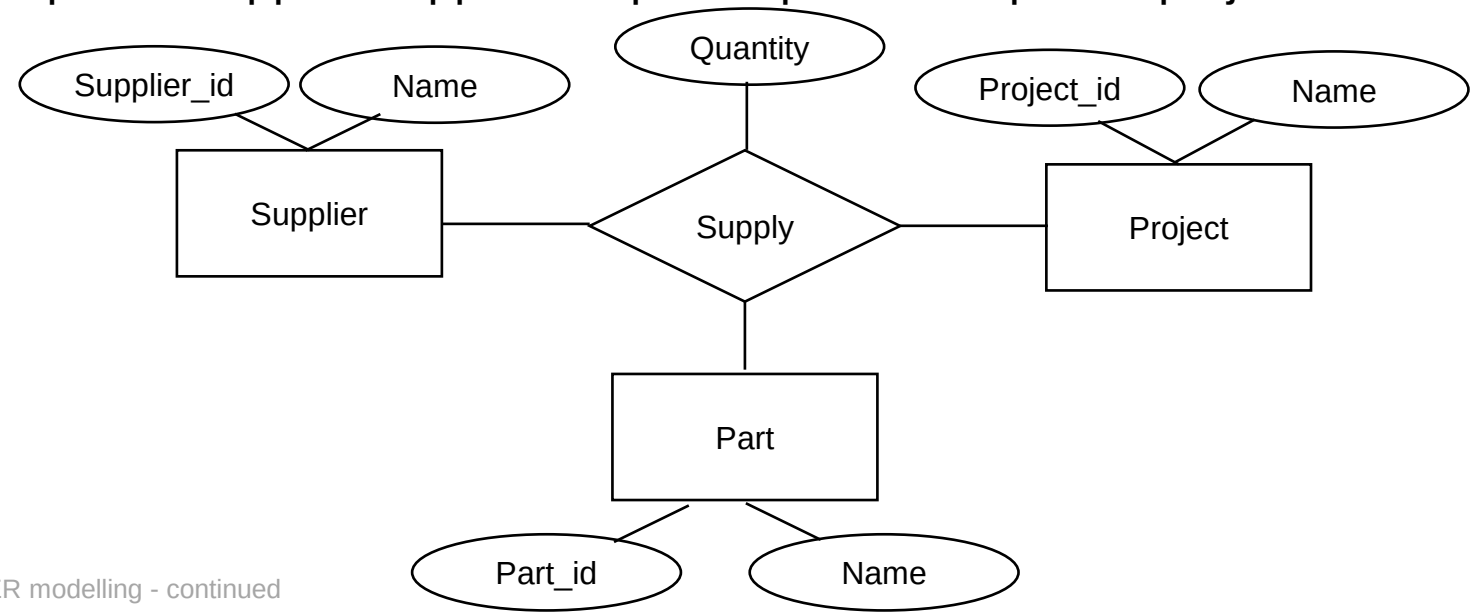
Composite attribute as key (3/3)

- Chain shop entities can be identified either by the shop ID or by the combination of the chain's name and location.

id	chain	location	street_address	postal_code	post_office
1	Chain X	Kaleva	A-street 22	33540	Tampere
2	Chain Y	Kaleva	A-street 22	33540	Tampere
3	Chain Y	Lielahti	B-street 1	33400	Tampere

N-ary relationship types (1/5)

- Earlier, unary and binary relationship types were discussed.
  - Unary: Associates entities of one entity type
  - Binary: Associates entities of two entity types
- A three-degree i.e. ternary relationship associates entities of three entity types.
- The Supply relationship type shown below describes a situation where a specific supplier supplies a specific part for a specific project.



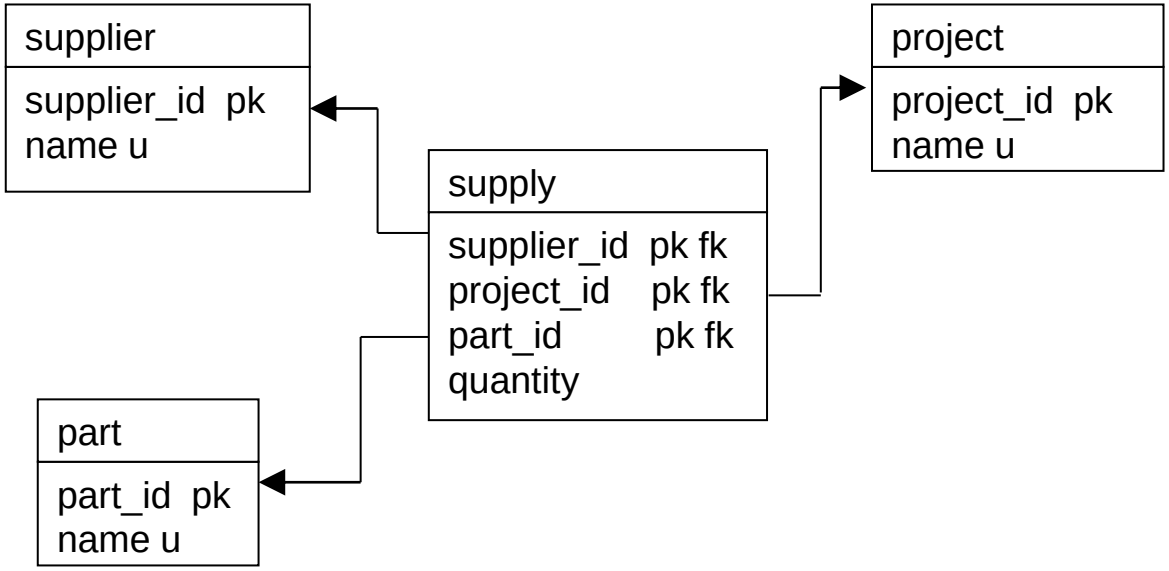
N-ary relationship types (2/5)

- To map ternary (or a higher-degree) relationship type, create a table for the relationship type.
- Include in the table as columns
  - the primary key columns of the tables of the entity types participating to the relationship type
    - Define foreign keys referencing to the corresponding tables
  - any simple attributes of the relationship type
- Define as the primary key of the relationship table the combination of the foreign key columns referencing to the entity type tables.

N-ary relationship types (3/5)

- The primary key of the supply table consists of three columns:

```
PRIMARY KEY (supplier_id, project_id, part_id)
```




```
CREATE TABLE supply(
supplier_id   INT NOT NULL,
project_id    INT NOT NULL,
part_id       INT NOT NULL,
quantity      INT NOT NULL,
PRIMARY KEY (supplier_id, project_id, part_id),
FOREIGN KEY (supplier_id) REFERENCES supplier,
FOREIGN KEY (project_id) REFERENCES project,
FOREIGN KEY (part_id) REFERENCES part);
```

N-ary relationship types (4/5)

- Supply relationships are identified by the triplet supplier_id, project_id and part_id.
- It is possible to represent the ternary relationship type also as a weak entity type with no partial key and with three identifying relationship types and three owner entity types.
  - In this case, we end up with the same scheme of the SQL database.

supplier

supplier_id	name
1	Bolt shop
2	Make's metal
3	Steel forge

project

project_id	name
1	Project A
2	Project B

part

part_id	name
1	Bolt 1
2	Bolt 2
3	Screw A
4	Screw B

supply

supplier_id	project_id	part_id	quantity
1	1	1	500
2	1	1	2000
3	2	4	1000

N-ary relationship types (5/5)

- Let us assume that there are trucks, products and shops as well as a ternary relationship type deliver. There are the following relationships
  - (t1,p1,s1)
  - (t2,p1,s2)
  - (t2,p2,s1)

  E.g. the truck 1 (t1) delivers the product 1 (p1) to the shop 1 (s1): (t1,p1,s1)

- Let's break down the ternary relationships into binary:
  - Relationships between trucks and products (t1,p1) (t2,p1) (t2,p2)
  - Relationships between trucks and shops (t1,s1) (t2,s2) (t2,s1)
  - Relationships between products and shops (p1,s1) (p1,s2) (p2,s1)

- Based on the binary relationships, one could erroneously conclude that the truck 2 delivers the product 1 for the shop 1. However, this is not the case - there is no triplet (t2,p1,s1).

- Therefore a ternary relationship is needed.

Keys

- A good primary key is a unique attribute or attribute combination whose value does not change (or changes very rarely).
  - A change in the value of the primary key may lead to a large number of updates.
- Sometimes the value of a natural attribute is unique and unchangeable (or rarely changing) e.g.
  - identity number, social security number
  - production number or serial number (e.g. cars, mobile phones)
- Usually artificial primary keys are used. Many times these are integer type.
  - For example, employee id as primary key and social security number as key (unique constraint).
- The ER schema must correspond to the SQL database schema. Therefore artificial keys are included in the ER diagram as well.

Design principles (1/2)

- Key principles of database design include avoiding storing the same data multiple times i.e. redundancy and avoiding missing values (NULL values).
  - Redundancy has the following problems:
    - Storing redundant data wastes storage space.
    - Maintaining redundant data needlessly consumes resources and is error-prone.
  - Missing values are associated with following problems:
    - Possible waste of storage space
    - NULL values must be taken into account in queries
    - Interpreting the meanings of the rows in the table may become more difficult
- All data in the database could be stored in a single table, but such a table would be problematic due to data redundancy and missing values.

Design principles (2/2)

- The aim is to design tables of “appropriate” size and avoid redundant data and missing values.

- When an ER schema is made and mapped into an SQL database schema according to the principles given, the process should result in a database with no useless redundancy and no useless missing values.
  - Sometimes, however, controlled redundancy is included in the database for efficiency and/or convenience of use.
  - In this case, it is necessary to make sure that the consistency of the data is maintained, i.e. that contradictory data cannot be stored in the database. This is a topic of later courses.

Material

- These slides are based on the book Elmasri, Navathe. Database systems: models, languages, design, and application programming. Pearson, 2011.