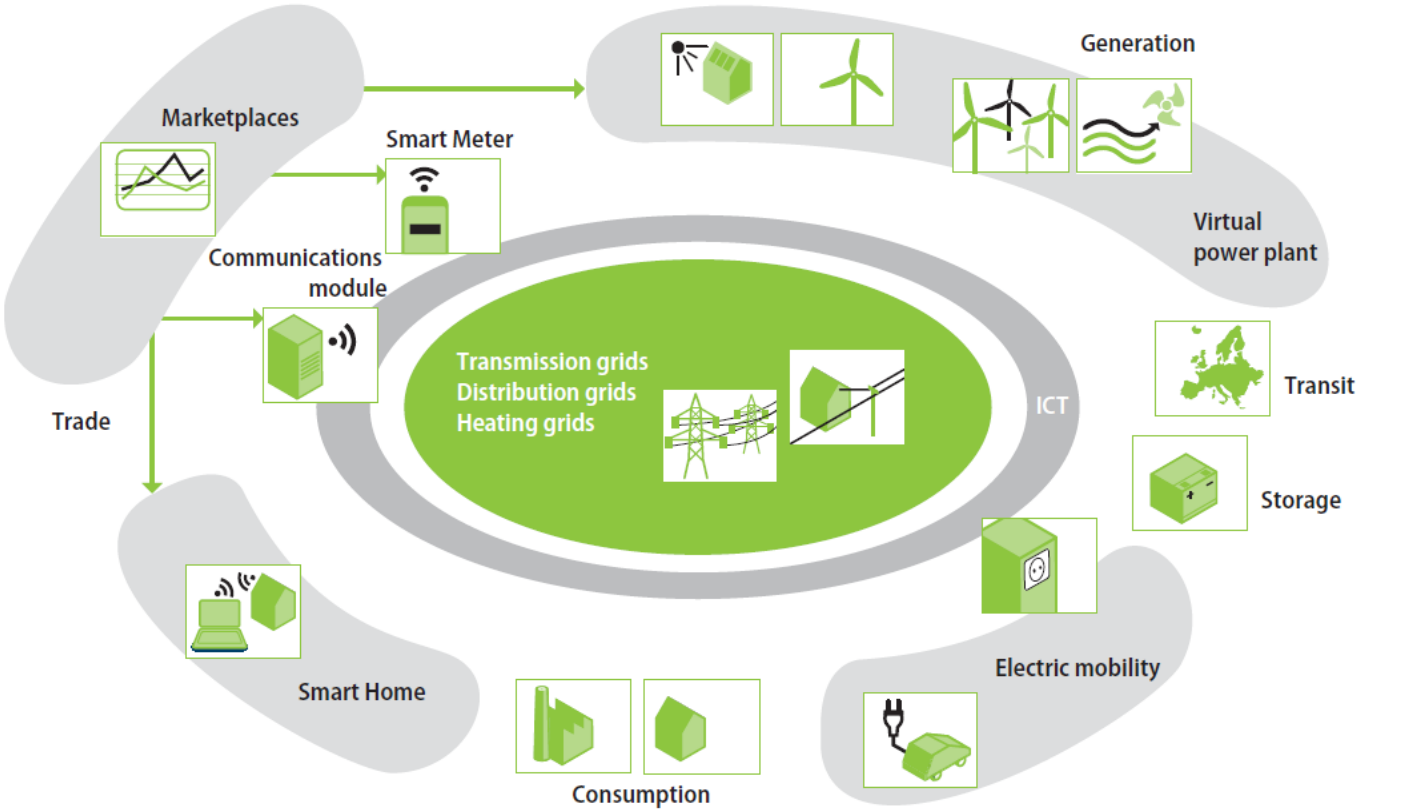
1. Motivation



Quelle: acatech, Future Energy Grid

Energy Management

- Synchronizing supply and demand
- Prognosis of supply and demand
- Load shifting
- Controlling power generating systems, managing storage devices
- Ancillary services for distributed resources: balancing power, reactive power,...
- Energy efficiency, user behavior

All these tasks are based on data.

Need to

- Model data,
- query and update data,
- react on data changes and model what has to happen,
- construct messages to communicate.

2. Data Modelling

Example

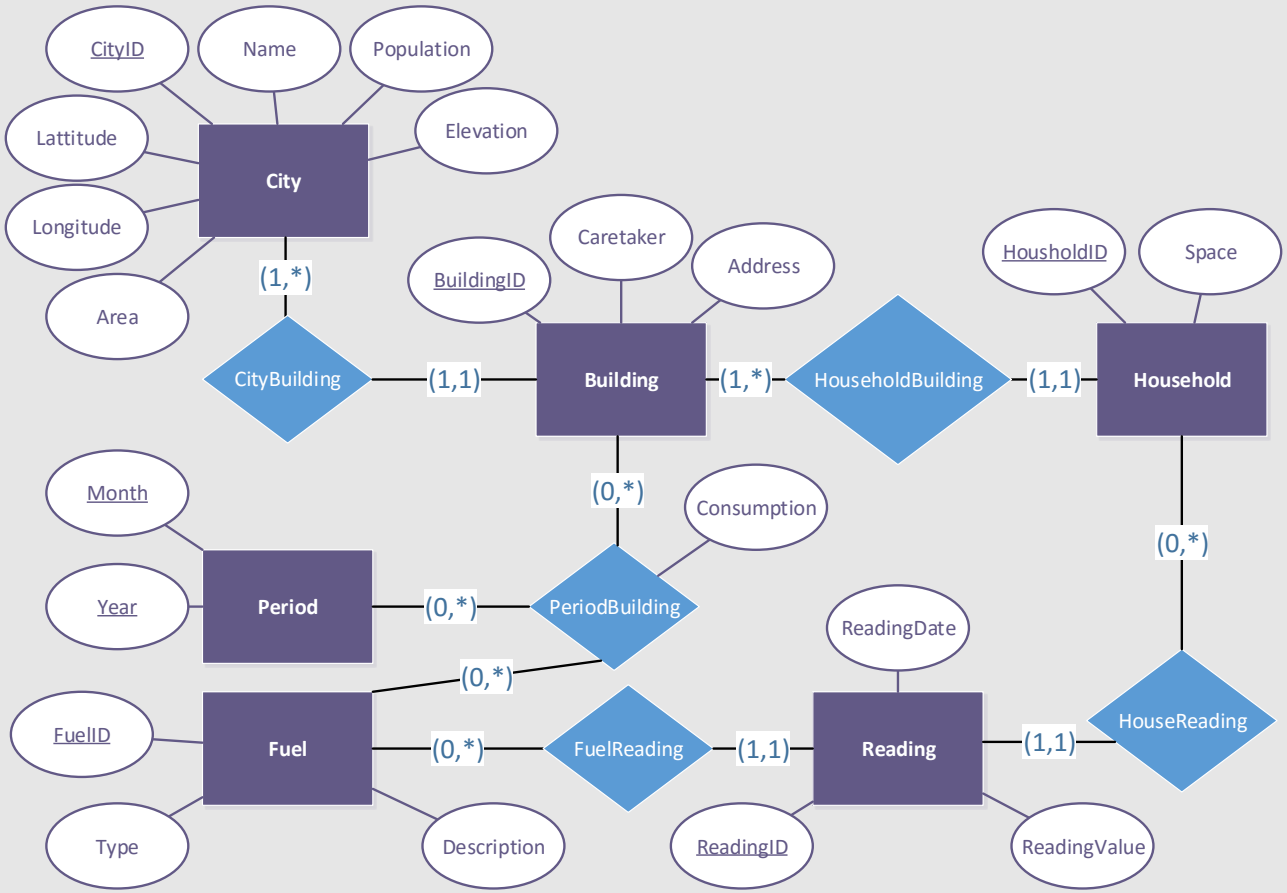
To be able to forecast energy consumption we maintain a database to record the consumption of cities households.

To this end we need data about cities, their buildings, the corresponding households and their energy consumption values taken as readings with respect to periods. To make the data and the relationships between data explicit we develop a model.

The model should show what we need to know about

- cities: *Name, Population, Area, Elevation, Latitude, Longitude*,
- buildings: *Address, Housekeeper*
- households: *Area*
- readings: *Date, Value, Fuel*.

Entity-Relationship Diagram (ER-Diagram, ERD)

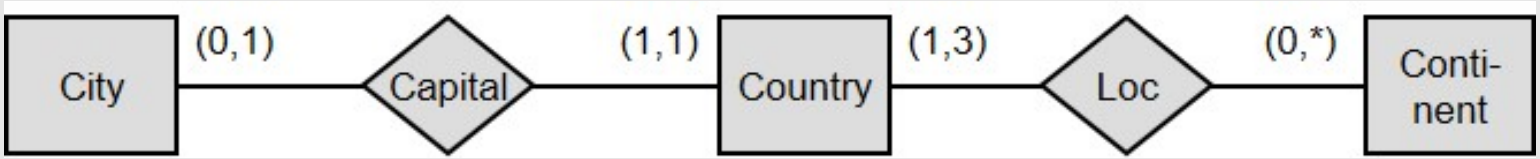


Entity-Sets are represented by rectangles, their *properties (attributes)* by ovals, *Relationship-Sets* by diamonds, which are further described by *cardinalities* attached to the connecting lines.

Entities and Relationships

- *Entities* must be uniquely identifiable by a *key*, i.e. a selected number of attributes graphically indicates by underlining. Typically the key is a certain artificial attribute, e.g. cityID, buildingID, etc. In general it may be defined by more than one attribute, e.g. to identify a building the name of the city and the address would suffice, as well.
- *Relationships* must be uniquely identifiable by the keys of the involved entities. Relationships may be defined over more than two entities, in general. Relationships over relationships are not allowed.

Example

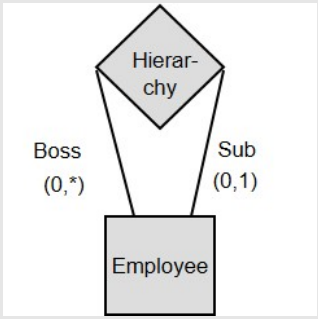


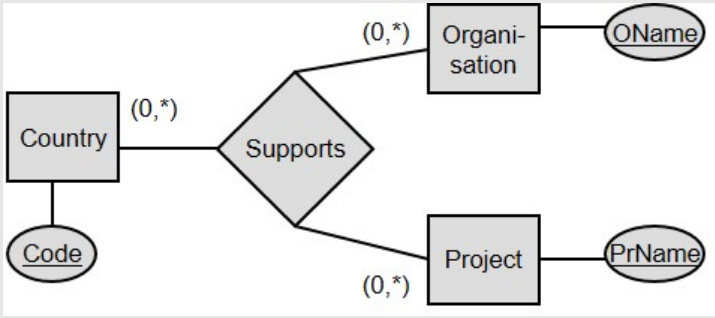
Cardinalities

- Let E —— R be an edge connecting, entity-set E and relationship-set R which is labelled by (min, max), $min \leq max$.
- (min, max) is called *cardinality* of E with respect to R.
- A *cardinality* (min, max) of E with respect to R states that each entity $e \in E$ is involved in at least *min* and at most *max* relationships $r \in R$.

  $*$ used for *max* means arbitrarily many.

Examples.



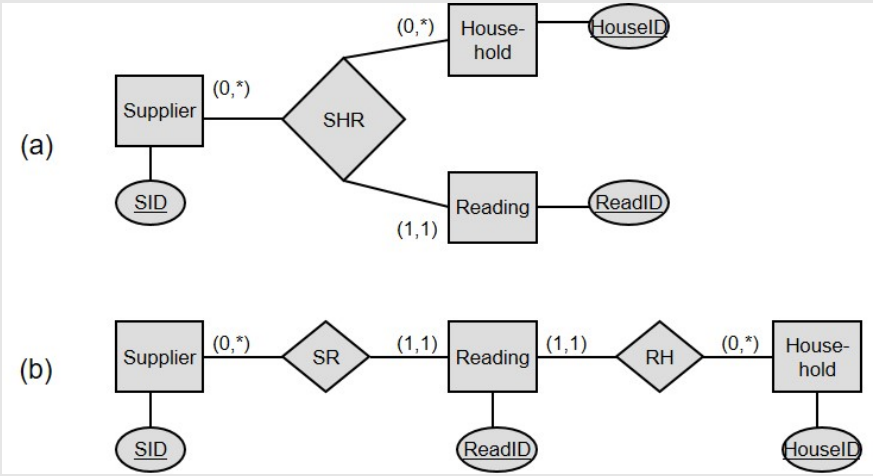


recursive relationships

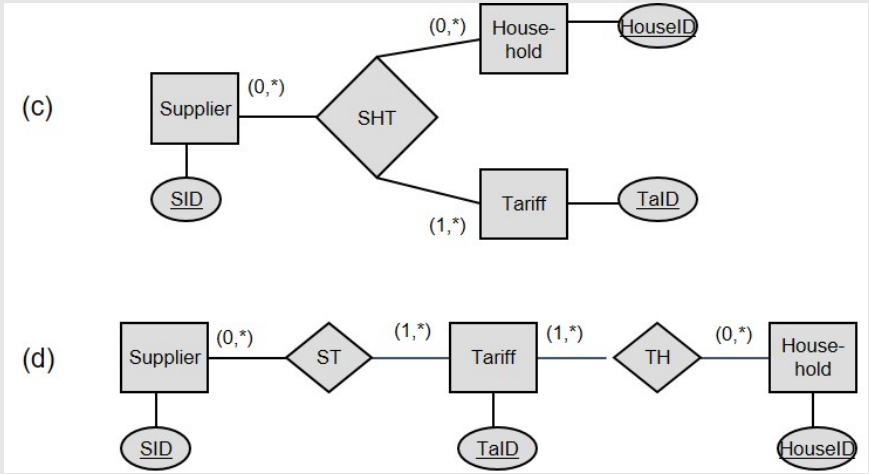
- An relationship-set is called *recursive*, whenever it is connected to the same relationship-set several times.

  For recursive relationship-sets we have to introduce *roles*.
- A relationship-set may be defined over more than two entity-sets.

Decomposition of relationship-sets.



(a) and (b) describe the same world;



(c) and (d) do not.

Example: why decomposition is not (always) allowed!

SHT

Supplier	Household	Tariff
SID	HouseID	TaID
Energiedienst	1020	Eco
Badenova	1020	Maxi
Badenova	1030	Eco

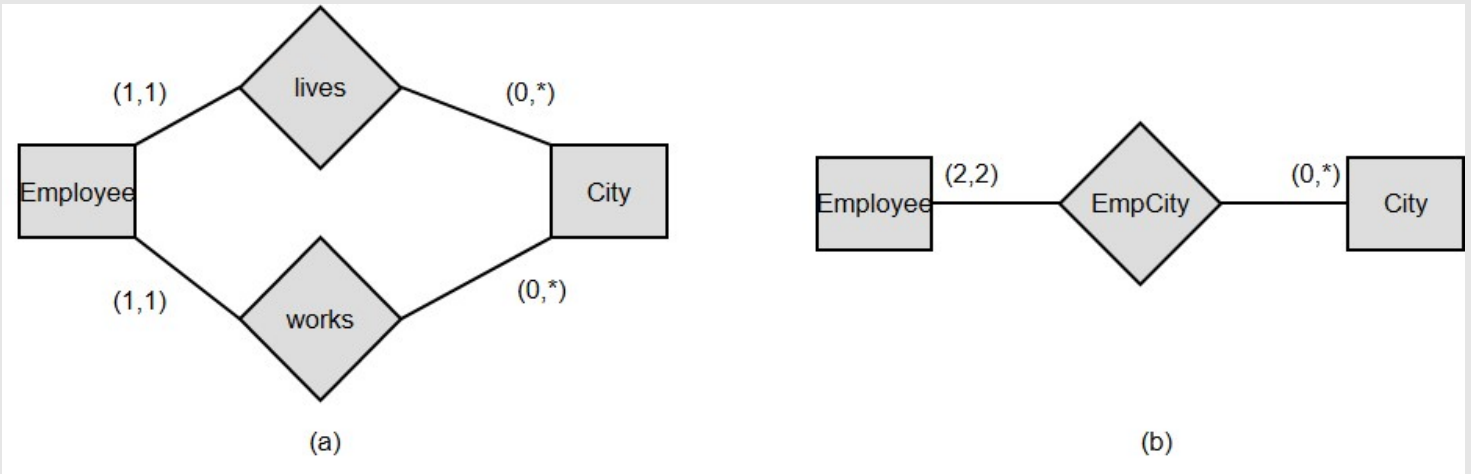
SO

Supplier	Tariff
SID	TaID
Energiedienst	Eco
Badenova	Maxi
Badenova	Eco

TH

Tariff	Household
TaID	HouseID
Eco	1020
Maxi	1020
Eco	1030

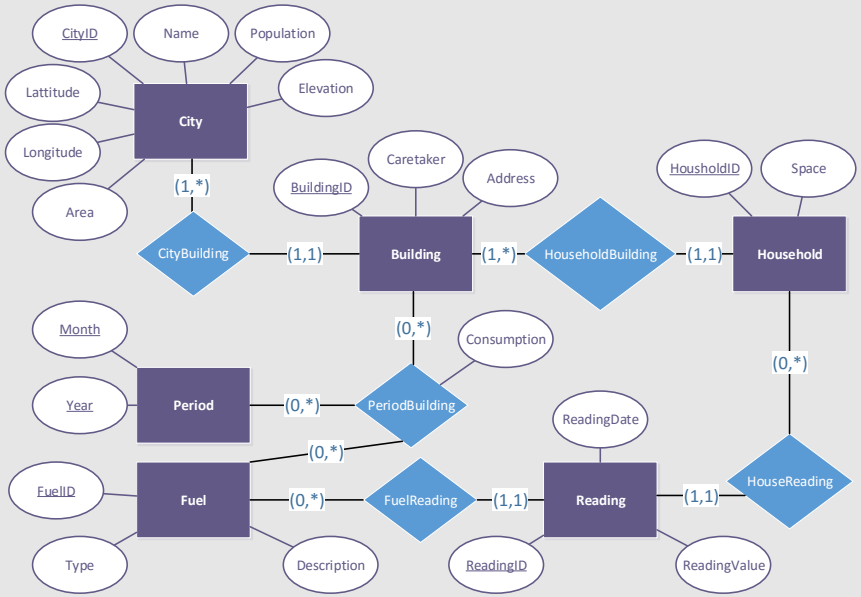
The same entity-sets may be involved in more than one relationship-set.



(a) and (b) describe different worlds.

3. Data Representation

Example: Mapping ER-Diagrams to Tables (Relations)



Tables for:

- `City`, with columns for `CityID`, `Name`, `Population`, `...`,
- `Building`, with columns for `BuildingID`, `CityID`, `Caretaker`, `Address`,
- etc.

Note how we treated relationship-set `CityBuilding` - it became part of table `Building`.This does not work for `PeriodBuilding`!However works for `HouseholdBuilding`, `HouseReading`, `FuelReading` analogously.

Definition of tables representing the information content modelled by the ER-Diagram

```
City(CityID, Name, Population Elevation, Lat, Long, Area)
Building(BuildinID, CityID, Caretaker, Address)
Household(HousID, BuildingID, Area)
Reading(ReadingID, HouseID, Date, Fuel, ReadinValus)
Fuel(FuelID, Type, Description)

Period(PeriodID, Year, Month)
PeriodBuilding(PeriodID, BuildingID, FuelID, Consumption)
```

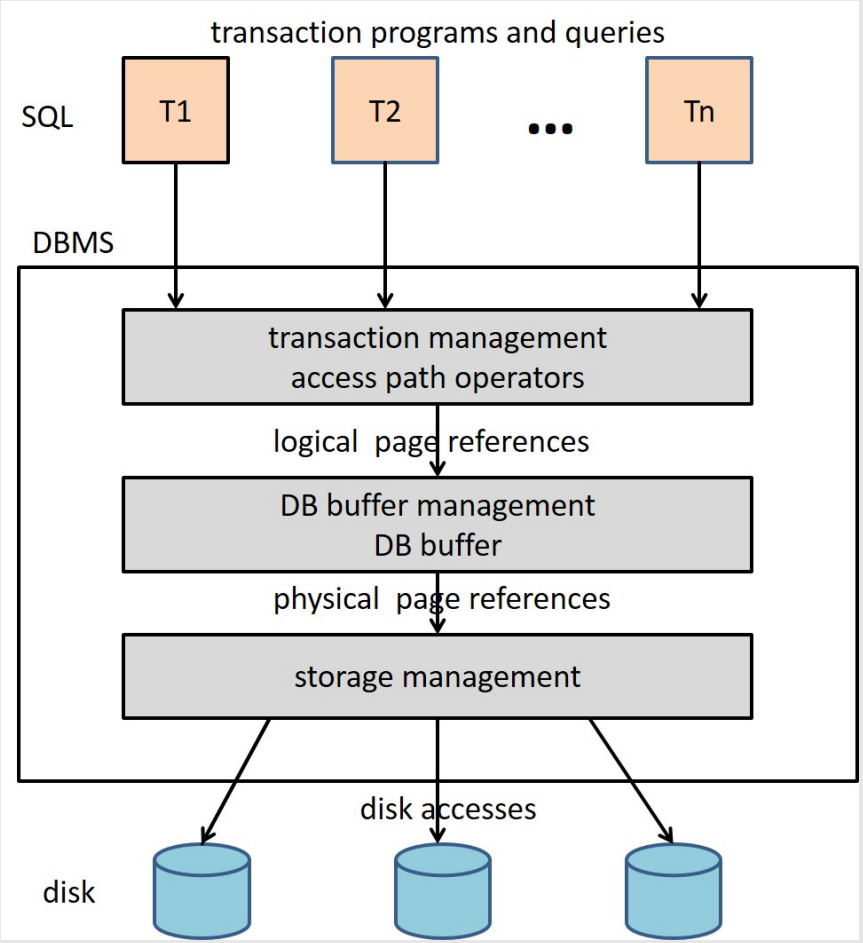
The rows of the tables contain the data - each column for each rows contains one value.

A system, which is able to process a set of tables, each containing a (very) large number of rows, is called *Relational Database System (DBS)*.

- To store, access and process the data managed by a DBS we need a language: SQL
- To make this working for a large number of users, a DBS provides a layered architecture: the user communicates with a *Data Base Management Software (DBMS)*, which organizes the access to the data efficiently.
- The data typically is stored on disks.

We concentrate on SQL.

Basic Architcture of a DBMS



4. SQL

http://dbissql.informatik.uni-freiburg.de/dbis/energy/sql.php

There you can work with a database of size:

Name of City	Buildings	Households	Readings
Freiburg im Breisgau	10877	97061	2329464
Karlsruhe	14446	128028	3072672
Kehl	1715	14827	355848
Stuttgart	29383	262710	6305040
Σ	56421	502626	12063024

Note: artificial numbers!

How to create a table:

```
CREATE TABLE City {
    CityID          NUMBER,
    Name            VARCHAR(80),
    Population      NUMBER,
    Area            NUMBER,
    Elevation       NUMBER,
    Latitude        NUMBER,
    Longitude       NUMBER,
PRIMARY KEY (CityID) };
```

The primary key guarantees unique references to the rows of the table.
Alternatively: `PRIMARY KEY (Latitude, Longitude)`.

How to avoid dangling references between tables:

```
CREATE TABLE Building {
    BuildingID      NUMBER,
    CityID          NUMBER,
    Address         VARCHAR(40),
    Caretaker       VARCHAR(40),
PRIMARY KEY (BuildingID),
FOREIGN KEY (CityID) REFERENCES City (CityID) };
```

The references clause guarantees that there will be no tuples in relation `Building`, for which the referenced city does not exist in table `City`.
The *referential integrity* is guaranteed. Later discussed: *referential actions*.

How to pose simple queries to a table:

- Give me all rows of a table.

```
SELECT * FROM City;
```

- Give me for all rows only the values of certain columns of a table.

```
SELECT CityID, Name, Area FROM City;
```

- Give me all (column values of) rows of a table which fulfill certain conditions.

```
SELECT CityID, Name, Area FROM City WHERE Area > 500;
```

- Give me all cities which are 'near' to Freiburg.

```
SELECT CityID, Name FROM City WHERE ??????????;
```

How to combine (join) tables:

- Give me a listing of building addresses with name of the city.

```
SELECT City.Name, Building.Address FROM City, Building
WHERE City.CityID = Building.CityID;
```

- Compute all pairs of cities.

```
SELECT A.Name, B.Name FROM City A, City B
WHERE A.CityID <> B.CityID;
```

- Give me all cities which are 'near' to Kehl.

```
SELECT B.Name FROM City A, City B
WHERE A.Name = 'Kehl' AND ABS(A.Latitude - B.Latitude) < 0.5
AND B.Name <> 'Kehl';
```

- Give me all the readings of Kehl nicely sorted.

```
SELECT D.HouseholdID, D.Fuel, D.ReadingValue
FROM City A, Building B, Household C, Reading D
WHERE A.Name = 'Kehl' AND A.CityID = B.CityID
AND B.BuildingID = C.BuildingID AND C.HousholdID = D.HousholdID
ORDER BY D.HouseholdIG ASC, D.Fuel ASC, D.ReadingValue DESC;
```