Tomas Sandnes
tomas.sandnes@kristiania.no

Week 9

DB1102 / PGR 111 – DATABASES

Today's topics

(Today's chapters: 8.2 in Norwegian book, 7.3 in English)

- Sum-up of ER modelling exercise.

- Normalization, part 1:
  - The purpose of normalization
  - Normalization terminology
  - The normalization process: UNF to 3NF

ER modelling

exercise sum-up

Exercise sum-up, ER modelling task

ER modelling, order of execution:

- Step 1: Create conceptual **model** by figuring out entities (tables), attributes (columns), and relationships.
  - *Note:* On this level, its ok to have M:M relationships.

- Step 2: Convert conceptual model to **table structure**: ("logical model")
  - Break up any M:M relationships by adding composite tables (entities).
  - Add primary keys: Exactly 1 to each table. (Note: FKs *can* span several columns.)
  - Add foreign keys: Exactly 1 to each relationship. (Note: For any 1:M relationship, *always on the "many" side!*) *Usually*, any FK matches the other side's PK.

- As an example, we look at Task 2 (part 1-5) form last week.

Normalization

part 1

The purpose of normalization

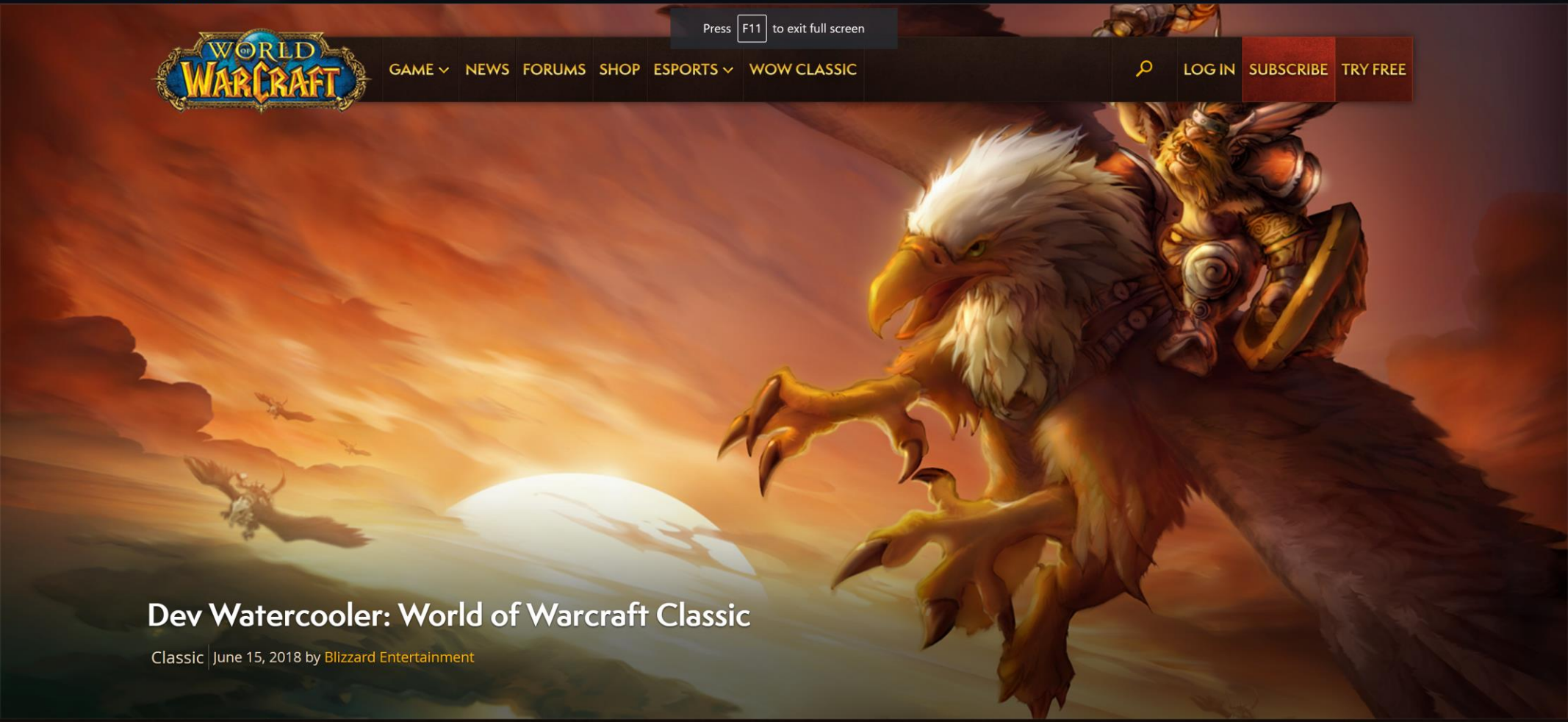
- The purpose of normalization is to find the most favorable relations for a given database.

- The main criterion we strive for in normalization:
  - Minimal double storage (redundancy), so attributes are only stored in one place.
  - Exception regarding Foreign Keys: These columns necessarily need to be stored in two places, since they are to connect tables (entities) through PK and FK.

- *Normalization usually means splitting the database into more tables.*

But is normalization used for anything IRL?

Oh yes! :-)



Dev Watercooler: World of Warcraft Classic

Classic | June 15, 2018 by Blizzard Entertainment

Example: Normalization in WoW (Classic)

- Was a very hot topic before WoW Classic launched! (August 2019.)
  - They wanted to use the modern WoW client.
  - But use the data (classes, abilities, items,...) from the original WoW version.

- Problem: The database had been normalized ("improved") in the meantime.
  - The data in the database (old, poorly normalized) did not match the client's expected input (modern, well-normalized format).

- Bottom line: *Git gud at normalization and get a job at Blizzard !? B-)*

- As mentioned earlier, to read more about WoW db-normalization, see here:
  - Dev Watercooler: World of Warcraft Classic

Case study: World of Warcraft – cont.

ID	Name	Effect One	Effect Two	Effect Three	Aura One	Aura Two	Effect Damage One	Aura Damage One	Aura Damage Two
1	Fireball	Deal Damage	Apply Aura	Nothing	Nothing	Deal Damage Periodically	30	Nothing	3
2	Frost Bolt	Deal Damage	Apply Aura	Nothing	Nothing	Slow	20	Nothing	Nothing

Poorly normalized

VS.

Well normalized

Table Name: Spell

ID	Name
1	Fireball
2	Frostbolt

+

Table Name: Spell Effect

ID	SpellID	Effect	Damage
1	1	Damage	30
2	2	Damage	25

+

Table Name: Spell Aura

ID	SpellID	Aura	Damage
1	1	Deal Damage Periodically	3
2	2	Slow	Nothing

Problems with double storage

- The table employee_branch (with primary key enr) contains double storage (redundancy) in several columns.
  - BNR, BADDRESS and BZIP: branch number, branch address and branch postal code.

ENR	NAME	EADDRESS	EZIP	POSITION	SALARY	BNR	BADDRESS	BZIP
3	Jon Hvit	Bruveien 7	4000	Manager	30000	1	Utleieveien 15	4000
4	Anne Strand	Strandgaten	2000	Broker	12000	1	Utleieveien 15	4000
20	Olav Gautesen	Galmannsveien 4	3000	Broker	26000	1	Utleieveien 15	4000
5	David Opalsen	Gulerleveien 43	2000	Secretary	18000	1	Utleieveien 15	4000
2	Marie Hovland	Strilegaten 8	5000	Manager	13000	2	Smuglerstien 67	5000
23	Ole Ås	Mor Åseveien 56	4000	Broker	17000	2	Smuglerstien 67	5000
21	Per Pollesen	Podlestadveien 5	5000	Secretary	15000	2	Smuglerstien 67	5000
7	Karl Hansen	Olavsgt 7	2000	Manager	25000	3	Snusveien 7	7000

- *What problems does this table give us, in terms of inserting, updating and deleting data?*

INSERT problems

- Inserting a new employee for an existing branch, problems:
  - Must at the same time re-enter data for a branch. (Branch number, address and postcode).
  - By typing these fields incorrectly, you get problems with the address being different for the same branch: you get an inconsistent database.

```
ENR NAME            EADDRESS         EZIP POSITION  SALARY BNR BADDRESS         BZIP
--- --------------- ---------------- ---- --------- ------ --- ---------------- ----
  3 Jon Hvit        Bruveien 7       4000 Manager    30000   1 Utleieveien 15   4000
  4 Anne Strand     Strandgaten      2000 Broker     12000   1 Utleieveien 15   4000
```

- Inserting a new branch without employees, problems:
  - Must enter NULL in the fields that apply to employees. But ENR is the PK and NULL is not allowed there.
  - Thus, must add a dummy employee.

DELETE and UPDATE problems

- Deleting last employee for a branch, problems:
  - At the same time loses all information about the branch.

```
ENR NAME            EADDRESS         EZIP POSITION  SALARY BNR BADDRESS         BZIP
--- --------------- ---------------- ---- --------- ------ --- ---------------- ----
  3 Jon Hvit        Bruveien 7       4000 Manager    30000   1 Utleieveien 15   4000
```

- Updating zip or address of a branch, problems:
  - Must make the same changes in all rows to all employees for this branch.
  - By typing incorrectly, you get problems with the address being different on the same branch: you get an inconsistent database.

```
ENR NAME            EADDRESS         EZIP POSITION  SALARY BNR BADDRESS         BZIP
--- --------------- ---------------- ---- --------- ------ --- ---------------- ----
  3 Jon Hvit        Bruveien 7       4000 Manager    30000   1 Utleieveien 15   4000
  4 Anne Strand     Strandgaten      2000 Broker     12000   1 Utleieveien 15   4000
```

Better table structure

- *Solution:* Split employee_branch into two tables – employee and branch.

ENR	NAME	EADDRESS	EZIP	POSITION	SALARY	BNR
3	Jon Hvit	Bruveien 7	4000	Manager	30000	1
4	Anne Strand	Strandgaten	2000	Broker	12000	1
20	Olav Gautesen	Galmannsveien 4	3000	Broker	26000	1
5	David Opalsen	Gulerleveien 43	2000	Secretary	18000	1
2	Marie Hovland	Strilegaten 8	5000	Manager	13000	2
23	Ole Ås	Mor Åseveien 56	4000	Broker	17000	2
21	Per Pollesen	Podlestadveien 5	5000	Secretary	15000	2
7	Karl Hansen	Olavsgt 7	2000	Manager	25000	3

BNR	BADDRESS	BZIP
1	Utleieveien 15	4000
2	Smuglerstien 67	5000
3	Snusveien 7	7000

- Are we now having problems with insert / update / delete?
  - No! :-) (More details on coming slide.)

Better table structure – cont.

- Inserting a new employee
  - Only enters personal information plus the correct branch number.
  - The branch address cannot be different for one and the same branch, as the branch information is only stored in one row in the database. (Has a consistent database).
- Inserting a new branch
  - It is perfectly okay that there is no information about employees in the branch table.
- Change of postcode or address of a branch
  - Only necessary to change the information for one row in the database. (Has a consistent database.)
- Delete last employee for a branch
  - The branch still exists in the branch table.

The normalization process

- Normalization usually means splitting the database into more tables.

- When a table is to be split into several new ones, it is important that:
  - We get a lossless split: All rows in the original table must be able to be formed by connecting rows in the new tables.
  - We preserve dependencies: constraints in the original table must be preserved by creating similar constraints in the new ones.

- We introduce the term functional dependency:
  - *Note:* We must understand the meaning (semantics) of the domain (database attributes) to be able to to do this process.
  - In other words: we must "understand the database" to be able to normalize it.

Definition, functional dependency

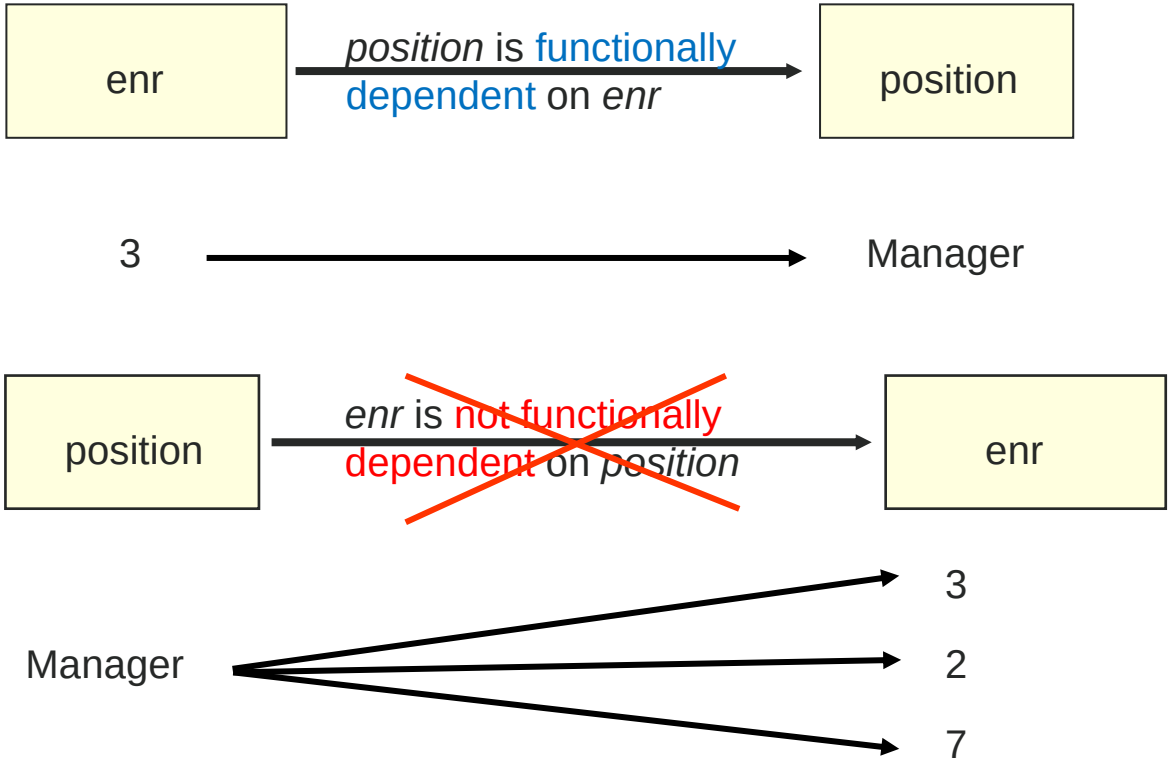
- If A and B are attributes in a relation R, then B is functionally dependent on A (can be written: A → B) if, for each value of A only one value of B can be identified, and if A is repeated, we also get the same value for B.
  - (A and B can consist of several attributes).

- *Short version, meaning of A → B:* If you have A, then you know (can find) B.

- Example: For our previous table, it can be said that '*position*' is functionally dependent on '*employee number*' ('employee number' → 'position'). The employee number must always refer to the same position.
  - The opposite is not necessarily the case: It is not necessarily a fixed employee number associated with each position. There can be several managers (each with their own employee number, i.e., different people) who lead their own department, etc.

- Slightly difficult concepts, but easier if we see a visual example. (Next slide.)

Example, functional dependency



Determinant

- Determinant:
  - one column (or more) in a table that determines at least one other column in the table.

- Generally:
  - If A → B (B functionally dependent on A) then A is determinant for B. (But B is not determinant for A.)

- The example on the previous page:
  - *enr* is determinant for *position*.
  - But *position* is not determinant for *enr*.

Question, functional dependency

- What functional dependencies do we find in the table below?

ENR	NAME	EADDRESS	EZIP	POSITION	SALARY	BNR	BADDRESS	BZIP
3	Jon Hvit	Bruveien 7	4000	Manager	30000	1	Utleieveien 15	4000
4	Anne Strand	Strandgaten	2000	Broker	12000	1	Utleieveien 15	4000
20	Olav Gautesen	Galmannsveien 4	3000	Broker	26000	1	Utleieveien 15	4000
5	David Opalsen	Gulerleveien 43	2000	Secretary	18000	1	Utleieveien 15	4000
2	Marie Hovland	Strilegaten 8	5000	Manager	13000	2	Smuglerstien 67	5000
23	Ole Ås	Mor Åseveien 56	4000	Broker	17000	2	Smuglerstien 67	5000
21	Per Pollesen	Podlestadveien 5	5000	Secretary	15000	2	Smuglerstien 67	5000
7	Karl Hansen	Olavsgt 7	2000	Manager	25000	3	Snusveien 7	7000

  - (The same table previously used in this lecture.)

Question, functional dependency – cont.

EMPLOYEE_BRANCH (enr, name, eaddress, ezip, position, salary, bnr, baddress, bzip)

enr → what is functionally dependent on *enr* ? → name, eaddress, ezip, position, salary, bnr, baddress, bzip

bnr → what is functionally dependent on *bnr* ? → baddress, bzip

Transitive dependency

- Transitive dependency describes (direct and) indirect functional dependency.
- A, B and C are attributes in a relationship.
- A → B and B → C.
- C is transitively dependent (*indirectly* functionally dependent) on A, via B.
  - (Its also directly functionally dependent on A.)

Question, transitive dependency

- What transitive dependencies can you find in the table below?

ENR	NAME	EADDRESS	EZIP	POSITION	SALARY	BNR	BADDRESS	BZIP
3	Jon Hvit	Bruveien 7	4000	Manager	30000	1	Utleieveien 15	4000
4	Anne Strand	Strandgaten	2000	Broker	12000	1	Utleieveien 15	4000
20	Olav Gautesen	Galmannsveien 4	3000	Broker	26000	1	Utleieveien 15	4000
5	David Opalsen	Gulerleveien 43	2000	Secretary	18000	1	Utleieveien 15	4000
2	Marie Hovland	Strilegaten 8	5000	Manager	13000	2	Smuglerstien 67	5000
23	Ole Ås	Mor Åseveien 56	4000	Broker	17000	2	Smuglerstien 67	5000
21	Per Pollesen	Podlestadveien 5	5000	Secretary	15000	2	Smuglerstien 67	5000
7	Karl Hansen	Olavsgt 7	2000	Manager	25000	3	Snusveien 7	7000

  - (Still the same table.)

Question, transitive dependency – cont.

ENR ----------> BNR

BNR -> BADDRESS, BZIP

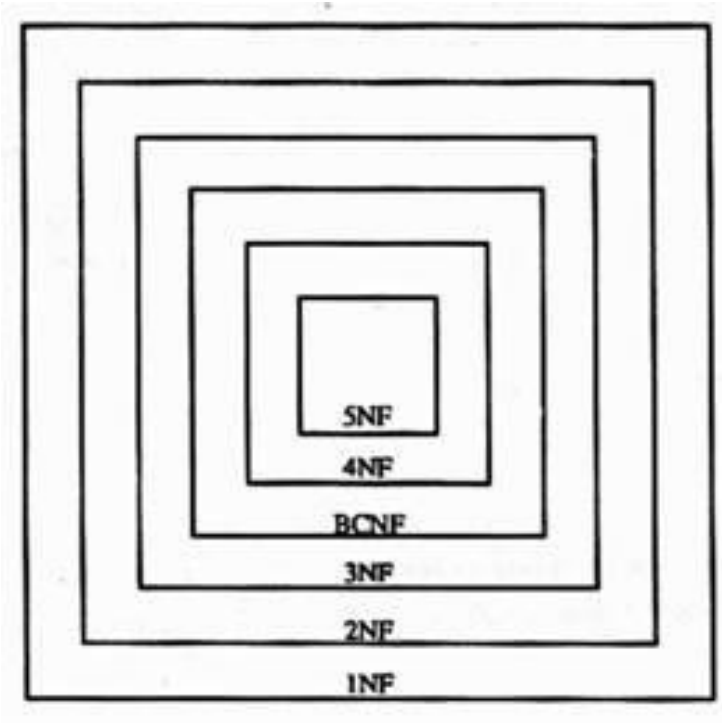
ENR ----------> BADDRESS, BZIP

ENR	NAME	EADDRESS	EZIP	POSITION	SALARY	BNR	BADDRESS	BZIP
3	Jon Hvit	Bruveien 7	4000	Manager	30000	1	Utleieveien 15	4000
4	Anne Strand	Strandgaten	2000	Broker	12000	1	Utleieveien 15	4000
20	Olav Gautesen	Galmannsveien 4	3000	Broker	26000	1	Utleieveien 15	4000
5	David Opalsen	Gulerleveien 43	2000	Secretary	18000	1	Utleieveien 15	4000
2	Marie Hovland	Strilegaten 8	5000	Manager	13000	2	Smuglerstien 67	5000
23	Ole Ås	Mor Åseveien 56	4000	Broker	17000	2	Smuglerstien 67	5000
21	Per Pollesen	Podlestadveien 5	5000	Secretary	15000	2	Smuglerstien 67	5000
7	Karl Hansen	Olavsgt 7	2000	Manager	25000	3	Snusveien 7	7000

`BADDRESS` og `BZIP` is transitively dependent on `ENR`, (indirectly dependent through `BNR`).

The normalization stages

- We'll now look at 1NF to 3NF. (First to third normal form.)
- Next time we'll look at BCNF. (Boyce-Codd normal form.)
- The other normal forms are not part of this subject.



Unnormalized form (UNF)

- New example: We register tenancy, consisting of: tenant (tnr, tname), property (pnr, paddress, rent), lease (fromdate, todate) and owner (onr, oname).
  - A table is unnormalized if it contains cells with more than one data element. (Cells that are not atomic.)

tnr	*tname*	*pnr*	*paddress*	*fromdate*	*todate*	*rent*	*onr*	*oname*
5	Hansen	6	Aveien 1	01-JUL-94	01-SEP-96	3500	4	Olsen
	?	1	Bveien 8	01-SEP-96	01-JAN-98	4400	12	Larsen
9	Persen	6	Aveien 1	01-SEP-96	01-SEP-97	3500	4	Olsen
	?	2	Dveien 6	01-SEP-97	01-SEP-98	5500	6	Alfsen
	?	5	Eveien 3	01-SEP-98	null	5000	6	Alfsen

First normal form (1NF)

- A table is 1NF if:
  - The cells contain only one data element.
  - (*Note:* "One data element" may very well be several words, for example in a varchar cell.)

- We can get rid of the problem with multiple data elements by creating one row per element:
  - Then each cell contains only one value.

Tenant_Lease_Property_Owner

tnr	tname	pnr	paddress	fromdate	todate	rent	onr	oname
5	Hansen	6	Aveien 1	01-JUL-94	01-SEP-96	3500	4	Olsen
5	*Hansen*	1	Bveien 8	01-SEP-96	01-JAN-98	4400	12	Larsen
9	Persen	6	Aveien 1	01-SEP-96	01-SEP-97	3500	4	Olsen
9	*Persen*	2	Dveien 6	01-SEP-97	01-SEP-98	5500	6	Alfsen
9	*Persen*	5	Eveien 3	01-SEP-98	null	5000	6	Alfsen

  - The primary key is here composed of: (tnr, pnr)

Second normal form (2NF)

- A table is 2NF if it is 1NF, and:
  - No *subset* of the PK is determinant for one or more other columns (no "partial dependency" on the PK).
  - *Note:* If a table is 1NF and the PK consists of only one column, then the table is also 2NF!

Ok: dependency on all parts of the PK.

tnr	tname	pnr	paddress	fromdate	todate	rent	onr	oname
5	Hansen	6	Aveien 1	01-JUL-94	01-SEP-96	3500	4	Olsen
5	Hansen	1	Bveien 8	01-SEP-96	01-JAN-98	4400	12	Larsen
9	Persen	6	Aveien 1	01-SEP-96	01-SEP-97	3500	4	Olsen
9	Persen	2	Dveien 6	01-SEP-97	01-SEP-98	5500	6	Alfsen
9	Persen	5	Eveien 3	01-SEP-98	null	5000	6	Alfsen

NOT ok: partial dependency on the PK.

Second normal form (2NF) – cont.

- We separate the columns that depend on parts of the primary key and place these in new tables:

Tenant

tnr	tname
5	Hansen
9	Persen

Lease

tnr	pnr	fromdate	todate
5	6	01-JUL-94	01-SEP-96
5	1	01-SEP-96	01-JAN-98
9	6	01-SEP-96	01-SEP-97
9	2	01-SEP-97	01-SEP-98
9	5	01-SEP-98	null

Property_Owner

pnr	paddress	rent	onr	oname
6	Aveien 1	3500	4	Olsen
1	Bveien 8	4400	12	Larsen
2	Dveien 6	5500	6	Alfsen
5	Eveien 3	5000	6	Alfsen

Third normal form (3NF)

- A table is 3NF if it is 2NF, and:
  - No non-PK attribute has a transitive dependency on a PK. (No A → B and B → C => A → C.)

- Alternatively, we can put it this way: All non-primary key attributes are only functionally dependent on the primary key.

```
pnr padresse leie onr oname
--- -------- ---- --- ------
  6 Aveien 1 3500   4 Olsen
  1 Bveien 8 4400  12 Larsen
  2 Dveien 6 5500   6 Alfsen
  5 Eveien 3 5000   6 Alfsen
```

Note: The other two tables, `Tenant` and `Lease`, are already at 3NF.

Third normal form (3NF) – cont.

- We remove transiently dependent attributes from the table and place them in their own table.

- The determinant becomes the PK of the new table.

Property

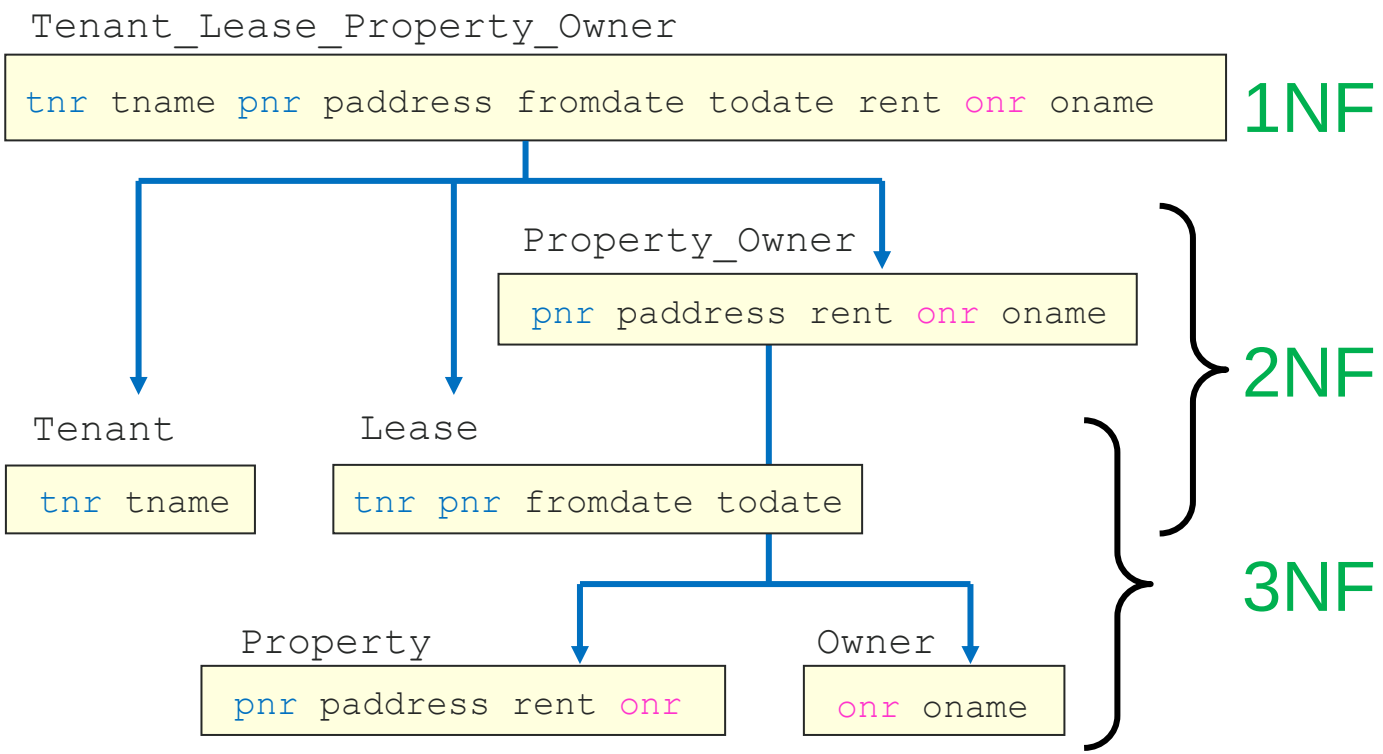
pnr	paddress	rent	onr
6	Aveien 1	3500	4
1	Bveien 8	4400	12
2	Dveien 6	5500	6
5	Eveien 3	5000	6

Owner

onr	oname
4	Olsen
12	Larsen
6	Alfsen

Done! :-)

Sum-up: 1NF to 3NF



1NF to 3NF, short version

- A table is 1NF if:
  - The cells contain only one data element. (Are atomic.)

- A table is 2NF if it is 1NF, and:
  - No subset of the PK is determinant for one or more other columns.
    (No "partial dependency" on the PK.)

- A table is 3NF if it is 2NF, and:
  - No non-PK attribute has a transitive dependency to a PK attribute.
    (No A → B and B → C => A → C.)

Today's exercises & looking ahead

- Now: 2 hours of exercises.

- Exercises are on Canvas, as usual. Short summary:
  - Tasks regarding normalization of databases.

- Main contents for the next lesson:
  - Normalization, part 2: Boyce-Codd normal form (BCNF).
  - (And, as a small topic, going the other way: Denormalization.)

The