Normalization
Database Examples

Database Normalization

- **1st Normal Form (1NF)**
- **There are no duplicated rows in the table.**
- **Each cell is single-valued (i.e., there are no repeating groups or arrays).**
- **Entries in a column (attribute, field) are of the same kind.**

Note:

The order of the rows is immaterial; the order of the columns is immaterial.

Note:

The requirement that there be no duplicated rows in the table means that the table has a key (although the key might be made up of more than one column—even, possibly, of all the columns).

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office #}

R1 = {Student ID, Last Name, First name}

R2 = {Student ID, Course ID, Course Section, Course Name, Grade}

R3 = {Course ID, Professor Last Name, Professor First Name, Bldg, Office#}

FD1 = Student ID -> Last Name, First Name

FD2 = Student ID -> Course ID, Course Section, Course Name, Grade

FD3 = Course ID -> Professor Last Name, Professor First Name, Bldg, Office#

Student ID -> Student ID (trivial)	F+ = {Student ID}
Student ID -> Last Name, First Name	F+ = {Student ID, Last Name, First Name}
Student ID -> Course ID	F+ = {Student ID, Last Name, First Name, Course ID}
Student ID -> C_Section, C_ Name	F+ = {Student ID, Last Name, First Name, C_ID, C_Section, C_Name}
Student ID -> Grade	F+ = {Student ID, Last Name, First Name, C_ID, C_Section, C_Name, Grade}
Course ID -> Prof_LName	F+ = {Student ID, Last Name, First Name, C_ID, C_Section, C_Name, Grade, Prof_ LName}
Course ID -> Prof_FName	F+ = {Student ID, Last Name, First Name, C_ID, C_Section, C_Name, Grade, Prog_ LName, Prof_FName}
Course ID -> Bldg, Office #	F+ = {Student ID, Last Name, First Name, C_ID, C_Section, C_Name, Grade, Prog_ LName, Prof_FName, Bldg, Office#}

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office#}

FD1 = Student ID --> Last Name, First Name

FD2 = Student ID --> Course ID, Course Section, Course Name, Grade

FD3 = Course ID --> Professor Last Name, Professor First Name, Bldg, Office#

R1 = {Student ID, Last Name, First name}

R1 is in First Normal Form

R2 = {Student ID, Course ID, Course Section, Course Name, Grade}

R2 is NOT in First Normal Form since Grade is a multi-valued attribute

Change FD2

FD2 = Student ID, Semester, Course ID, Course Section --> Course Name, Grade

R3 = {Course ID, Professor Last Name, Professor First Name, Bldg, Office#}

R3 is NOT in First Normal Form since Professor Last Name is a multi-valued attribute

Change FD3

FD3 = Semester, Course ID, Course Section --> Professor Last Name, Professor First Name, Bldg, office#

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office #, Semester}

FD1 = Student ID --> Last Name, First Name

FD2 = Student ID, Semester, C_ID, C_Section --> C_Name, Grade

FD3 = Semester, C_ID, C_Section --> Prof_LName, Prof_FName, Bldg, Office#

Student

Student_ID	L_ Name	F_Name
001	Smith	John
002	Smith	Susan
003	Beal	Fred
004	Thomson	Marie

Courses

Student_ID	Semester	C_ID	C_Section	CName	Grade
001	Fall03	Eng01	1N	English	A
001	Spr04	Ger01	2N	German	B
002	Fall03	Eng01	1N	English	A
002	Spr04	Ger01	2N	German	B
003	Spr04	Ger01	1N	German	A
004	Spr04	Ita01	1N	Italian	B

Faculty

Semester	C_ID	C_Section	Prof_LName	Prof_FName	Bldg	Office#
Fall03	Eng01	1N	Ruger	John	LG	102
Spr04	Eng01	1N	Ruger	John	LG	102
Spr04	Ger01	1N	Findling	Holger	LG	101
Spr04	Ger01	1N	Findling	Holger	LG	101
Spr04	Ital01	2N	Fresco	Luise	LG	103

2nd Normal Form (2NF)

- **Def: A table is in 2NF if it is in 1NF and if all non-key attributes are dependent on all of the key.**

Note:

Since a partial dependency occurs when a non-key attribute is dependent on only a part of the (composite) key, the definition of 2NF is sometimes phrased as, "A table is in 2NF if it is in 1NF and if it has no partial dependencies."

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office #, Semester}

FD1 = Student ID --> Last Name, First Name

FD1 is in 1st and 2nd normal form

FD2 = Student ID, Semester, C_ID, C_Section --> C_Name, Grade

FD2 is not in 2nd normal form, since C_Name is not functionally dependent on Student ID

Change to

FD2 = Student ID, Semester, C_ID, C_Section → Grade

FD4 = Course ID → Course Name

FD3 = Semester, C_ID, C_Section --> Prof_LName, Prof_FName, Bldg, Office#

FD3 is in 2nd normal form, even so the non-key attributes are not represented well in this relation.

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office #, Semester}

FD1 = Student ID --> Last Name, First Name

FD2 = Student ID, Semester, C_ID, C_Section → Grade

FD4 = Course ID → Course Name

FD3 = Semester, C_ID, C_Section --> Prof_LName, Prof_FName, Bldg, Office#

Student

Student_ID	L_ Name	F_Name
001	Smith	John
002	Smith	Susan
003	Beal	Fred
004	Thomson	Marie

Courses

Student_ID	Semester	C_ID	C_Section	Grade
001	Fall03	Eng01	1N	A
001	Spr04	Ger01	2N	B
002	Fall03	Eng01	1N	A
002	Spr04	Ger01	2N	B
003	Spr04	Ger01	1N	A
004	Spr04	Ita01	1N	B

Course Name

C_ID	CName
Eng01	English
Ger01	German
Ita01	English

Faculty

Semester	C_ID	C_Section	Prof_LName	Prof_FName	Bldg	Office#
Fall03	Eng01	1N	Ruger	John	LG	102
Spr04	Eng01	1N	Ruger	John	LG	102
Spr04	Ger01	1N	Findling	Holger	LG	101
Spr04	Ger01	1N	Findling	Holger	LG	101
Spr04	Ital01	2N	Fresco	Luise	LG	103

3rd Normal Form (3NF)

- **Def: A table is in 3NF if it is in 2NF and if it has no transitive dependencies.**

Boyce-Codd Normal Form (BCNF)

- **Def: A table is in BCNF if it is in 3NF and if every determinant is a candidate key.**

Database Normalization

FD1 = Student ID --> Last Name, First Name

FD2 = Student ID, Semester, C_ID, C_Section → Grade

FD4 = Course ID → Course Name

FD3 = Semester, C_ID, C_Section --> Prof_LName, Prof_FName, Bldg, Office#

FD3 is not in 3rd nomal form, since Bldg and Office# is transitive to Prof_LName, Prof_FName

Change FD3

FD3 = Semester, C_ID, C_Section --> Faculty ID

FD5 = Faculty ID → Prof_LName, Prof_FName, Bldg, Office#

Database Normalization

R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg, Office #, Semester, Faculty_ID}

FD1 = Student ID --> Last Name, First Name

FD2 = Student ID, Semester, C_ID, C_Section → Grade

FD4 = Course ID → Course Name

FD3 = Semester, C_ID, C_Section --> Faculty ID

FD5 = Faculty ID → Prof_LName, Prof_FName, Bldg, Office#

Student

Student_ID	L_ Name	F_Name
001	Smith	John
002	Smith	Susan
003	Beal	Fred
004	Thomson	Marie

Courses

Student_ID	Semester	C_ID	C_Section	Grade
001	Fall03	Eng01	1N	A
001	Spr04	Ger01	2N	B
002	Fall03	Eng01	1N	A
002	Spr04	Ger01	2N	B
003	Spr04	Ger01	1N	A
004	Spr04	Ita01	1N	B

Course Name

C_ID	CName
Eng01	English
Ger01	German
Ita01	English

Faculty

Semester	C_ID	C_Section	Faculty_ID
Fall03	Eng01	1N	001
Spr04	Eng01	1N	001
Spr04	Ger01	1N	002
Spr04	Ger01	1N	002
Spr04	Ital01	2N	003

SemesterStaff

Faculty_ID	Prof_LName	Prof_FName	Bldg	Office#
001	Ruger	John	LG	102
002	Findling	Holger	LG	101
003	Fresco	Luise	LG	103

Boyce-Codd Normal Form

- A relation schema *R* is in **Boyce-Codd Normal Form (BCNF)** with respect to a set *F* of functional dependencies if for all functional dependencies in F+ of the form a → B, where a is a proper subset of R and B is a proper subset of R, at least one of the following holds:
  - a → B is a trivial functional dependency (i.e. B is proper subset of a).
  - a is a superkey for schema *R*.

- A database design is in BCNF if each member of the set of relation schemas is in BCNF.

Let's assess our example banking design:

- *Customer-schema = (cname, street, ccity)*
- *cname* → *street ccity*

- *Branch-schema = (bname, assets, bcity)*
- *bname* → *assets bcity*

- *Loan-info-schema = (bname, cname, loan#, amount)*
- *loan#* → *amount bname*

- *Customer-schema* and *Branch-schema* are in BCNF.

Boyce-Codd Normal Form

Let's look at *Loan-info-schema*:

- We have the non-trivial functional dependency *loan#* → *amount*, and
- *loan#* is not a superkey.
- Thus *Loan-info-schema* is not in BCNF.
- We also have the repetition of information problem.
- For each customer associated with a loan, we must repeat the branch name and amount of the loan.
- We can eliminate this redundancy by decomposing into schemas that are all in BCNF.

If we decompose into

- *Loan-schema = (bname, loan#, amount)*
- *Borrow-schema = (cname, loan#)*
- we have a lossless-join decomposition. (Remember why?)
- To see whether these schemas are in BCNF, we need to know what functional dependencies apply to them.
- For *Loan-schema*, we have *loan#* → *amount bname* applying.
- Only trivial functional dependencies apply to *Borrow-schema*.
- Thus both schemas are in BCNF.
- We also no longer have the repetition of information problem. Branch name and loan amount information are not repeated for each customer in this design.

Comparison of BCNF and 3NF

- We have seen BCNF and 3NF.
  - It is always possible to obtain a 3NF design without sacrificing lossless-join or dependency-preservation.
  - If we do not eliminate all transitive dependencies, we may need to use null values to represent some of the meaningful relationships.
  - Repetition of information occurs.
- These problems can be illustrated with *Banker-schema*.
  - As *banker-name bname* , we may want to express relationships between a banker and his or her branch.

cname	banker-name	bname
Bill	John	SFU
Tom	John	SFU
Mary	John	SFU
null	Tim	Austin

Figure 7.4: An instance of *Banker-schema*.

  - Figure 7.4 shows how we must either have a corresponding value for customer name, or include a null.
  - Repetition of information also occurs.
  - Every occurrence of the banker's name must be accompanied by the branch name.

If we must choose between BCNF and dependency preservation, it is generally better to opt for 3NF.

- If we cannot check for dependency preservation efficiently, we either pay a high price in system performance or risk the integrity of the data.
- The limited amount of redundancy in 3NF is then a lesser evil.

- To summarize, our goal for a relational database design is
  - BCNF.
  - Lossless-join.
  - Dependency-preservation.

- If we cannot achieve this, we accept
  - 3NF
  - Lossless-join.
  - Dependency-preservation.

- **A final point:** there is a price to pay for decomposition. When we decompose a relation, we have to use natural joins or Cartesian products to put the pieces back together. This takes computational time.

4th Normal Form (4NF)

- **Def: A table is in 4NF if it is in BCNF and if it has no multi-valued dependencies.**