IT 775
Database Technology

SQL

Normalization

Database Normalization

The main goal is to restructure the logical data model of a database to:

Eliminate redundancy

Organize data efficiently

Reduce the potential for data anomalies

The Process of Normalization

As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

Data Anomalies

Data anomalies are inconsistencies in the data stored in a database as a result of an operation such as update, insertion, and/or deletion.

Such inconsistencies may arise when have a particular record stored in multiple locations and not all of the copies are updated.

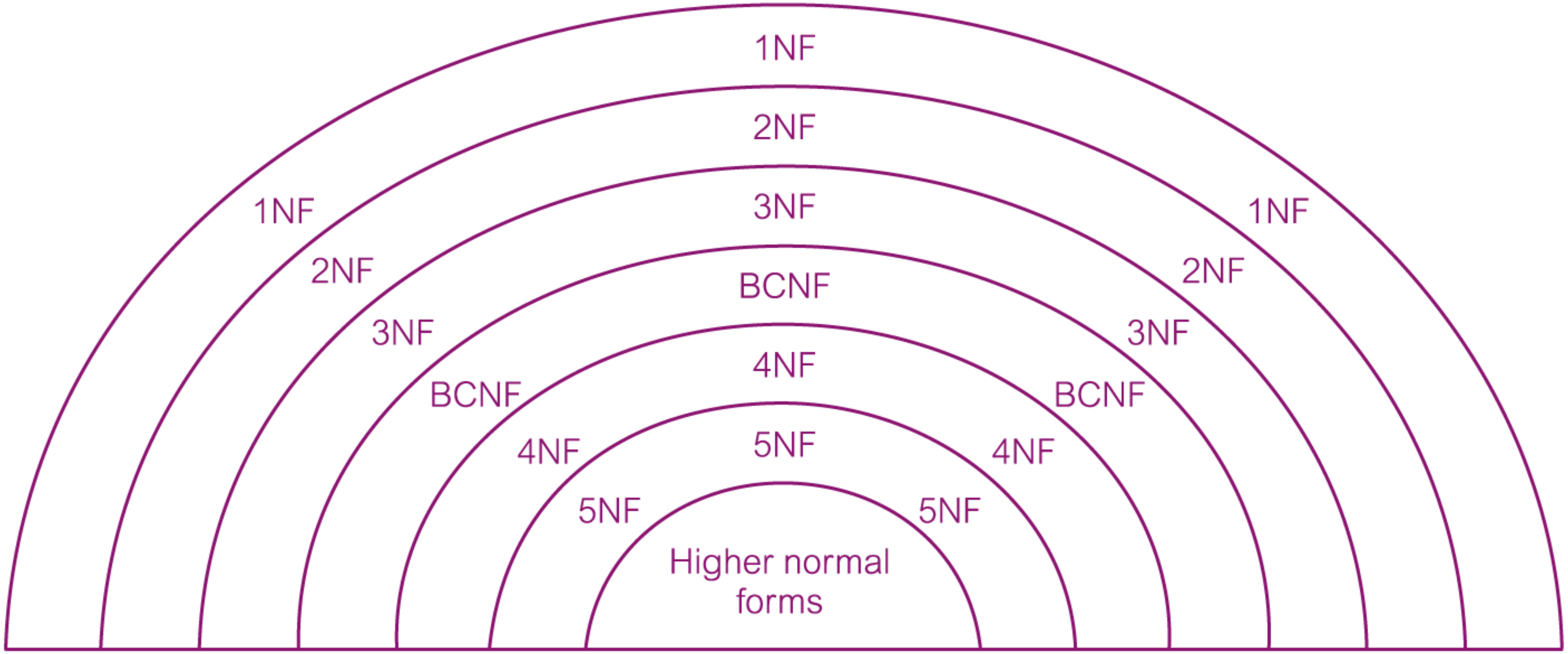
We can prevent such anomalies by implementing 7 different level of normalization called Normal Forms (NF)

Brief History/Overview

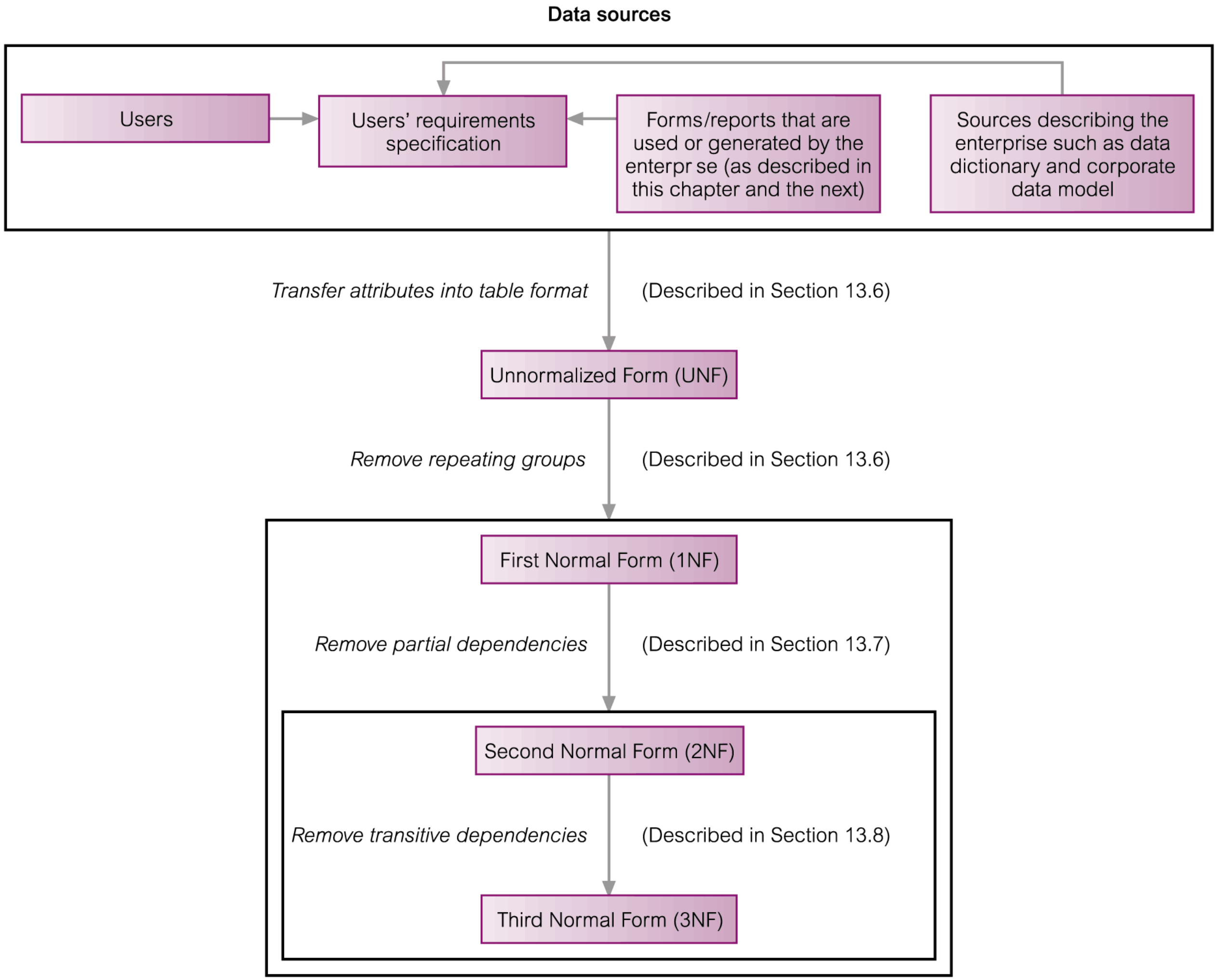
Database Normalization was first proposed by Edgar F. Codd. Codd defined the first three of the seven known Normal Forms.

Normal Forms are progressive. In order to have 3rd NF, we must have 2nd NF, and in order to have 2nd NF, we must have 1st NF.

The Process of Normalization



The Process of Normalization



Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterpr se (as described in this chapter and the next)
Sources describing the enterprise such as data dictionary and corporate data model
Transfer attributes into table format
(Described in Section 13.6)
Unnormalized Form (UNF)
Remove repeating groups
(Described in Section 13.6)
First Normal Form (1NF)
Remove partial dependencies
(Described in Section 13.7)
Second Normal Form (2NF)
Remove transitive dependencies
(Described in Section 13.8)
Third Normal Form (3NF)

The Process of Normalization

The normalization process involves examining each table and verifying if it satisfies a particular normal form

If a table satisfies a particular normal form, then the next step is to verify if that relation satisfies the next higher normal form

If a table does not satisfy a particular normal form, actions are taken to convert the table into a set of tables that satisfy the particular normal form

The Process of Normalization

Normalizing to *first normal form* is done on non-relational tables in order to convert them to relational tables

Normalizing to *subsequent normal forms* (e.g., second normal form, third normal form) improves the design of relational tables that contain redundant information and alleviates the problem of update anomalies

NORMALIZATION

There are several normal forms, most fundamental of which are:

First normal form (1NF)

Second normal form (2NF)

Third normal form (3NF)

Unnormalized Form (UNF)

A table that contains one or more repeating groups.

To create an unnormalized table:

Transform the data from the information source (e.g. form) into table format with columns and rows.

First Normal Form (1NF)

A relation in which the intersection of each row and column contains one and only one value.

The requirements to satisfy the 1st NF:

Each table has a primary key: a minimal set of attributes which can uniquely identify a record

The values in each column of a table are atomic (No multi-value attributes allowed).

There are no repeating groups: two columns do not store similar information in the same table.

UNF to 1NF

Nominate an attribute or group of attributes to act as the key for the unnormalized table.

Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).

UNF to 1NF

Remove the repeating group by:

Entering appropriate data into the empty columns of rows containing the repeating data (‘flattening’ the table).

Or by:

Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

1st Normal Form -- Example

Un-normalized Student table:

Student#	AdvID	AdvName	AdvRoom	Class1	Class2
123	123A	James	555	102-8	104-9
124	123B	Smith	467	209-0	102-8

Normalized Student table:

Student#	AdvID	AdvName	AdvRoom	Class#
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

Second Normal Form (2NF)

Based on the concept of full functional dependency

Full functional dependency indicates that if

A and B are attributes of a relation,

B is fully dependent on A if B is functionally dependent on A, but not on any proper subset of A.

The requirements to satisfy the 2nd NF:

All requirements for 1st NF must be met.

Redundant data across multiple rows of a table must be moved to a separate table.

- The resulting tables must be related to each other by use of foreign key.

Second Normal Form (2NF)

A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.

1NF to 2NF

Identify the primary key for the 1NF relation.

Identify the functional dependencies in the relation.

If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.

2nd Normal Form -- Example

Normalized (1NF) Student table:

Student#	AdvID	AdvName	AdvRoom	Class#
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

2nd Normal Form -- Example

Student table:

Student#	AdvID	AdvName	AdvRoom
123	123A	James	555
124	123B	Smith	467

Registration table:

Student#	Class#
123	102-8
123	104-9
124	209-0
124	102-8

Third Normal Form (3NF)

Based on the concept of transitive dependency

Transitive Dependency is a condition where

A, B and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).

The requirements to satisfy the 3rd NF:

All requirements for 2nd NF must be met.

Eliminate fields that do not depend on the primary key;

- That is, any field that is dependent not only on the primary key, but also on another field, must be moved to another table.

Third Normal Form (3NF)

A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.

2NF to 3NF

Identify the primary key in the 2NF relation.

Identify functional dependencies in the relation.

If transitive dependencies exist on the primary key, remove them by placing them in a new relation along with a copy of their dominant.

3rd Normal Form -- Example

Students (2NF) table:

Student#	AdvID	AdvName	AdvRoom
123	123A	James	555
124	123B	Smith	467

Student table:

Student#	AdvID
123	123A
124	123B

Advisor table:

AdvID	AdvName	AdvRoom
123A	James	555
123B	Smith	467

3rd Normal Form -- Example Cont.

Student table:

Student#	AdvID
123	123A
124	123B

Registration table:

Student#	Class#
123	102-8
123	104-9
124	209-0
124	102-8

Advisor table:

AdvID	AdvName	AdvRoom
123A	James	555
123B	Smith	467

SQL Criteria -- Normal Forms

First Normal Form (1NF):

Table must be two-dimensional, with rows and columns.

Each row contains data that pertains to one item or one portion of an item.

Each column contains data for a single attribute of the item being described.

Each cell (intersection of row and column) of the table must be single-valued.

All entries in a column must be of the same kind.

Each column must have a unique name.

No two rows may be identical.

The order of the columns and of the rows does not matter.

Second Normal Form (2NF):

Table must be in first normal form (1NF).

All non-key attributes (columns) must be dependent on the entire key.

Third Normal Form (3NF):

Table must be in second normal form (2NF).

Table has no transitive dependencies.

NORMALIZATION

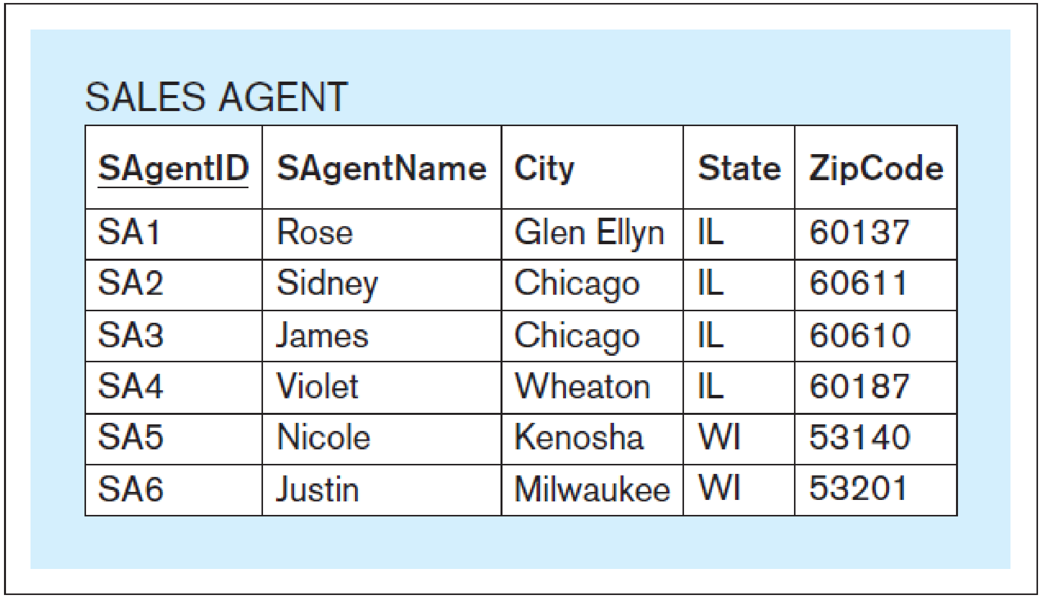
Normalization Exceptions

In general, database relations are normalized to 3NF in order to eliminate unnecessary data redundancy and avoid update anomalies.

However, normalization to 3NF should be done judiciously and pragmatically, which may in some cases call for deliberately not normalizing certain relations to 3NF.

Normalization exception example: relation SALES AGENT

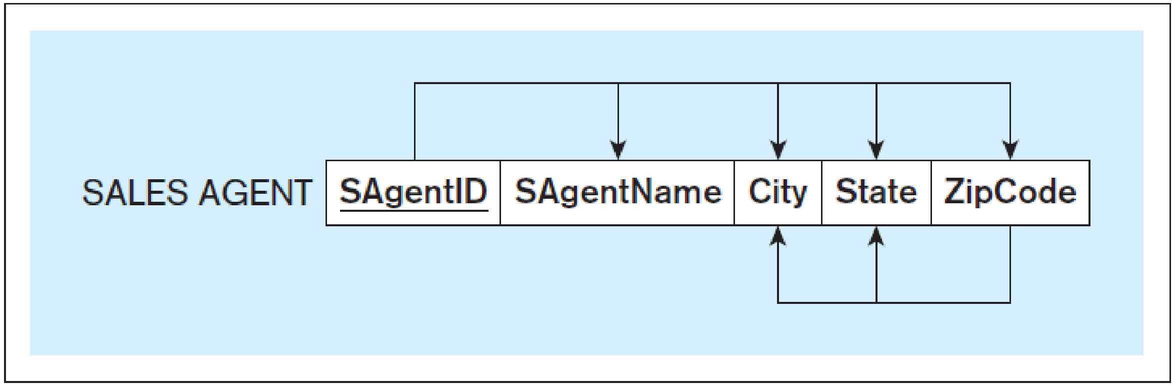
Relation SALES AG



SALES AGENT

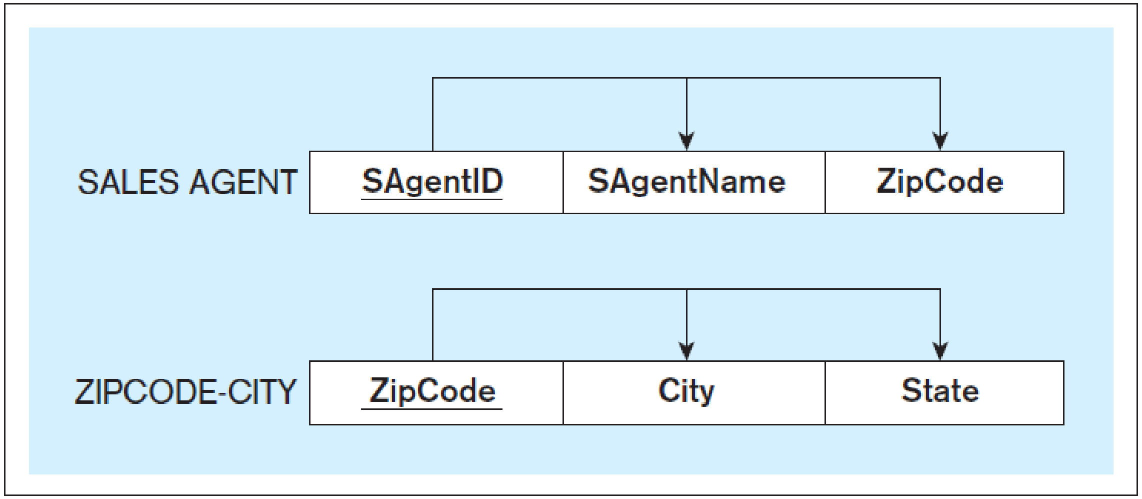
SAgentID	SAgentName	City	State	ZipCode
SA1	Rose	Glen Ellyn	IL	60137
SA2	Sidney	Chicago	IL	60611
SA3	James	Chicago	IL	60610
SA4	Violet	Wheaton	IL	60187
SA5	Nicole	Kenosha	WI	53140
SA6	Justin	Milwaukee	WI	53201

Functional dependencies in the relation SALES AGENT



Normalization exception example: should relation SALES AGENT be normalized?

SALES AGENT example in 3NF



SALES AGENT

<u>SAgentID</u>	SAgentName	ZipCode

ZIPCODE-CITY

<u>ZipCode</u>	City	State

NORMALIZATION

Denormalization - reversing the effect of normalization by joining normalized relations into a relation that is not normalized, in order to improve query performance

The data that resided in fewer relations prior to normalization is spread out across more relations after normalization

This has an effect on the performance of data retrievals

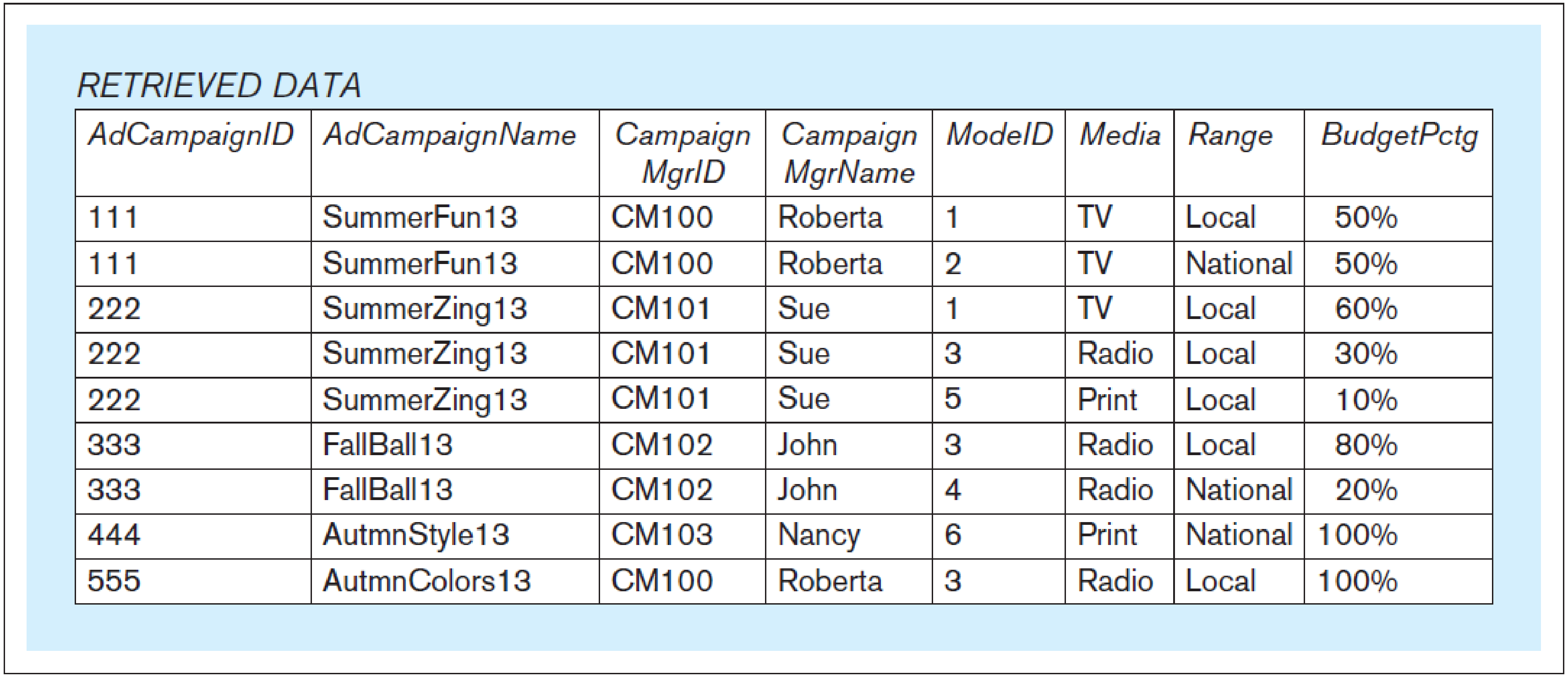
Denormalization can be used in dealing with the normalization vs. performance issue

Denormalization is not a default process that is to be undertaken in all circumstances

Instead, denormalization should be used judiciously, after analyzing its costs and benefits

Denormalization example

Pressly Ad Agency example—a retrieval of data



RETRIEVED DATA

AdCampaignID	*AdCampaignName*	*Campaign MgrID*	*Campaign MgrName*	*ModelD*	*Media*	*Range*	*BudgetPctg*
111	SummerFun13	CM100	Roberta	1	TV	Local	50%
111	SummerFun13	CM100	Roberta	2	TV	National	50%
222	SummerZing13	CM101	Sue	1	TV	Local	60%
222	SummerZing13	CM101	Sue	3	Radio	Local	30%
222	SummerZing13	CM101	Sue	5	Print	Local	10%
333	FallBall13	CM102	John	3	Radio	Local	80%
333	FallBall13	CM102	John	4	Radio	National	20%
444	AutmnStyle13	CM103	Nancy	6	Print	National	100%
555	AutmnColors13	CM100	Roberta	3	Radio	Local	100%

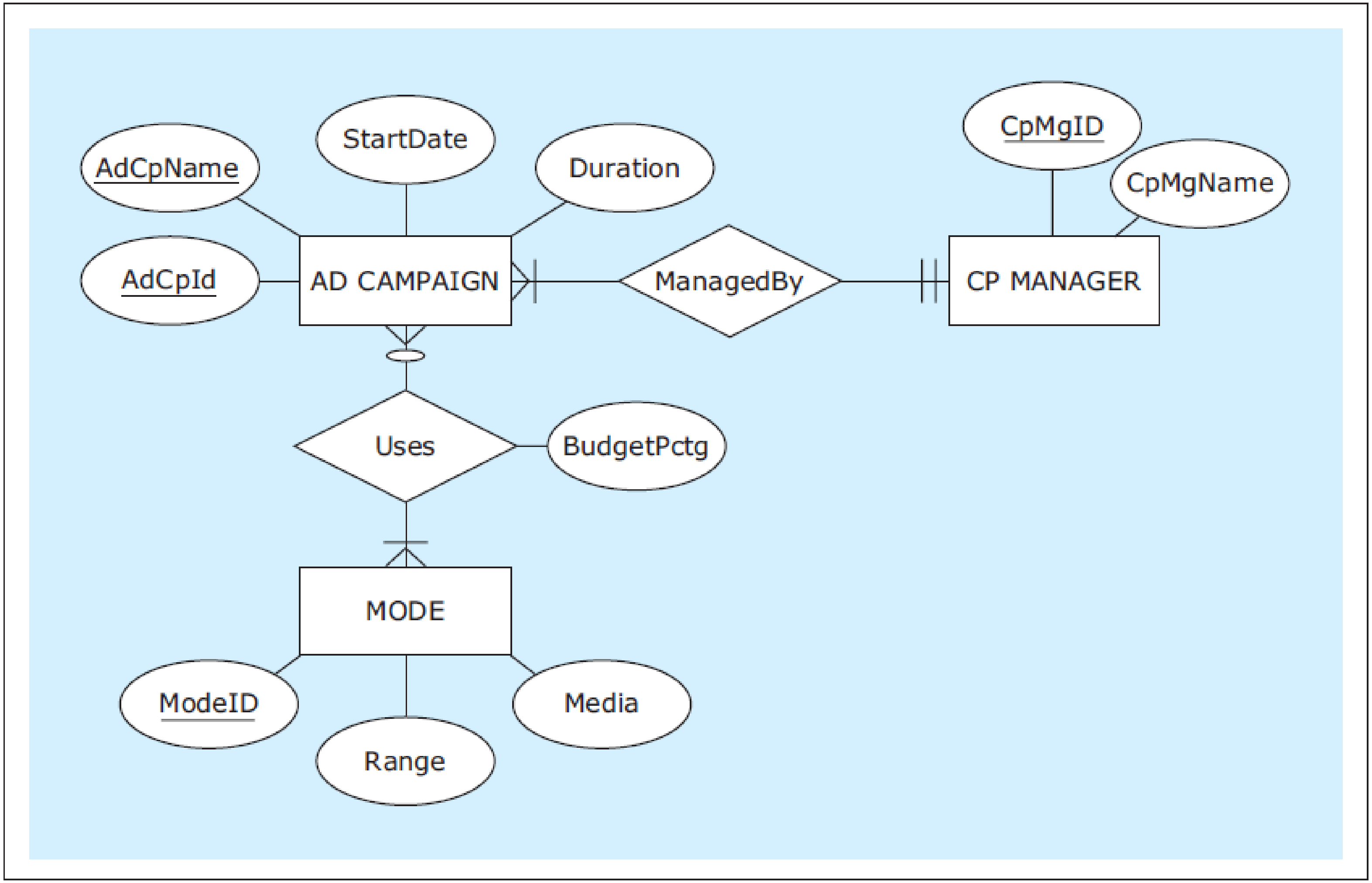
NORMALIZATION

ER-Modeling versus Normalization

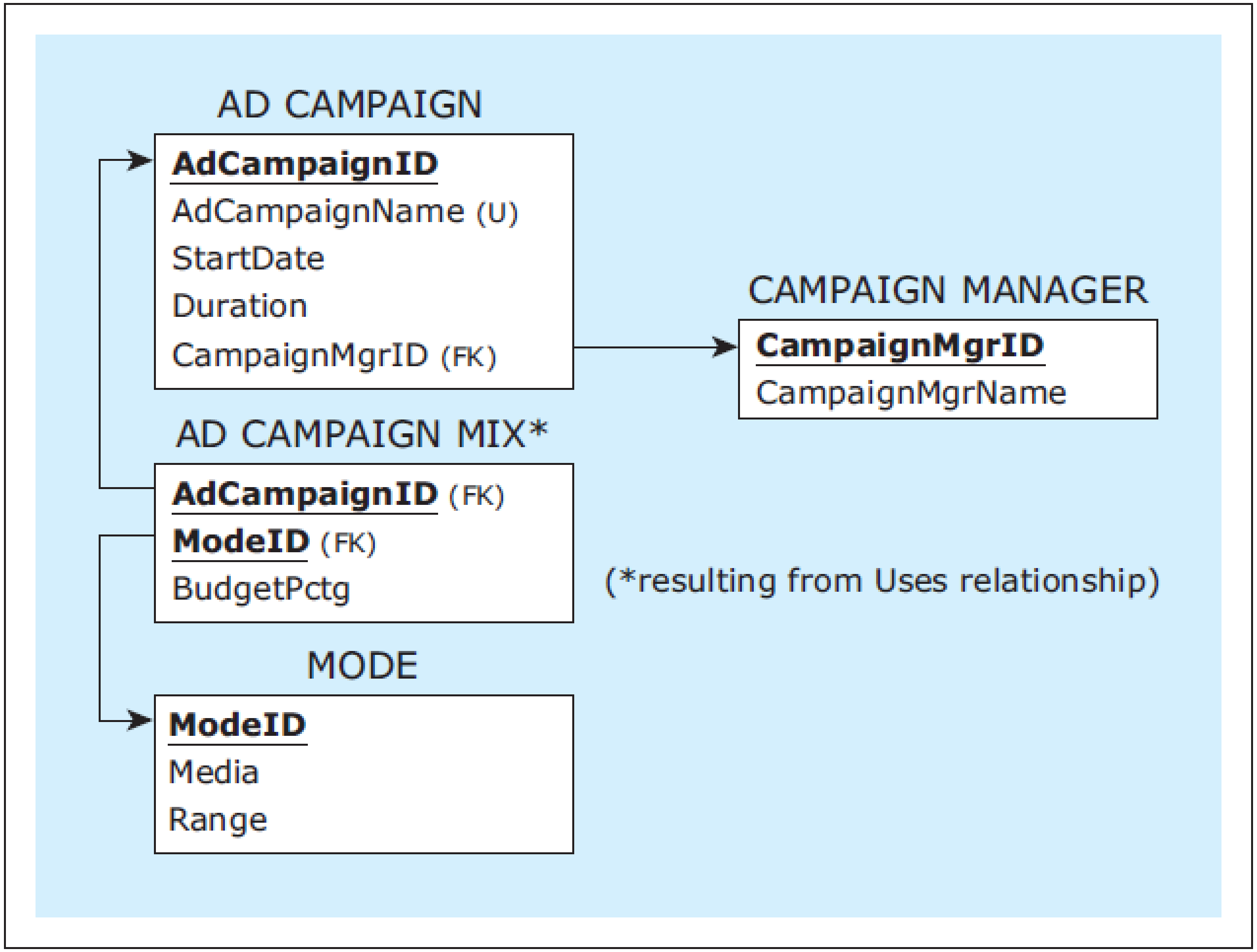
ER modeling followed by mapping into a relational schema is one of the most common database design methods.

When faced with a non-normalized table, instead of identifying functional dependencies and going through normalization to 2NF and 3NF, a designer can analyze the table and create an ER Diagram based on the table (and subsequently map the table into a relational schema)

ER-Modeling vs. Normalization example - ER diagram of the Pressly Ad Agency



ER-Modeling vs. Normalization example - mapped ER diagram, identical to the relational schema resulting from the normalization process (the relational schema of 3NF relations for the normalized Pressly Ad Agency example)



AD CAMPAIGN

AdCampaignID
AdCampaignName (U)
StartDate
Duration
CampaignMgrID (FK)

CAMPAIGN MANAGER

CampaignMgrID
CampaignMgrName

AD CAMPAIGN MIX*

AdCampaignID (FK)
ModeID (FK)
BudgetPctg

(*resulting from Uses relationship)

MODE

ModeID
Media
Range

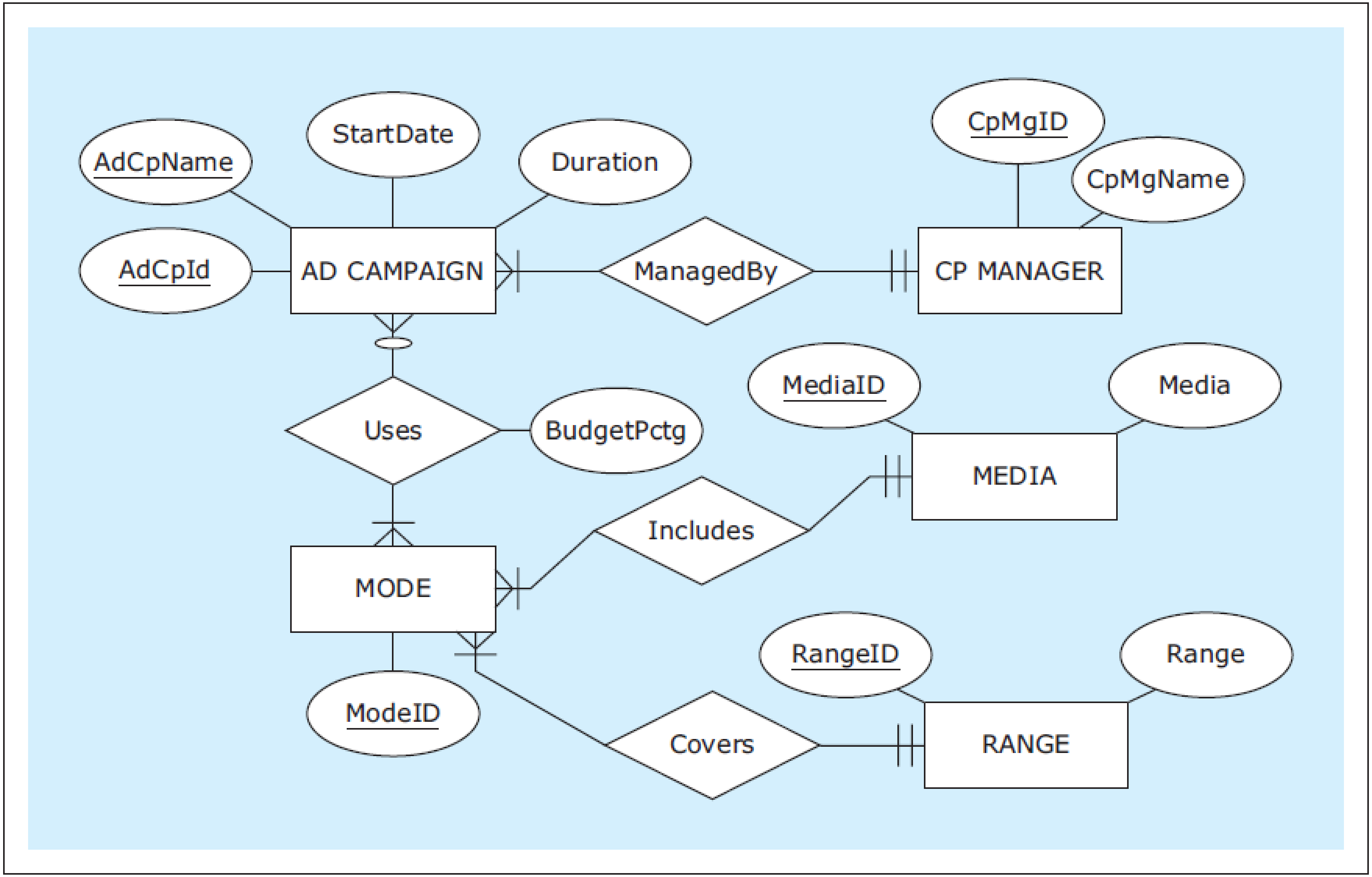
ADDITIONAL STREAMLINING OF DATABASE CONTENT

Designer-added entities (tables) and keys

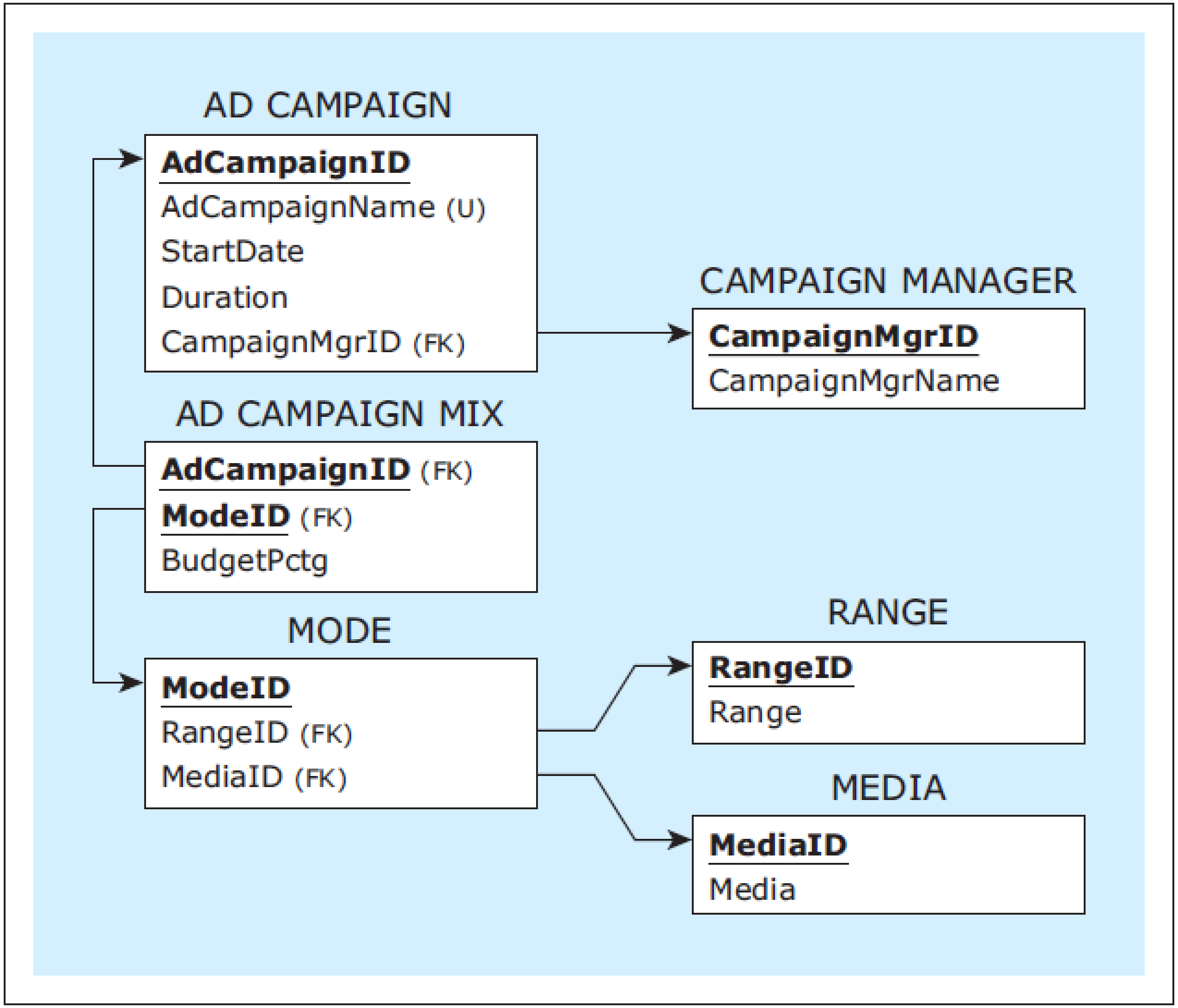
Even if a relation is in 3NF additional opportunities for streamlining database content may still exist.

Designer-added entities (tables) and designer-added keys can be used for additional streamlining.

Designer-added entities (tables) and keys example - augmented ER diagram of the Pressly Ad Agency example



Designer-added entities (tables) and keys example - augmented ER diagram of the Pressly Ad Agency example mapped into a relational schema



AD CAMPAIGN

AdCampaignID
AdCampaignName (U)
StartDate
Duration
CampaignMgrID (FK)

CAMPAIGN MANAGER

CampaignMgrID
CampaignMgrName

AD CAMPAIGN MIX

AdCampaignID (FK)
ModeID (FK)
BudgetPctg

MODE

ModeID
RangeID (FK)
MediaID (FK)

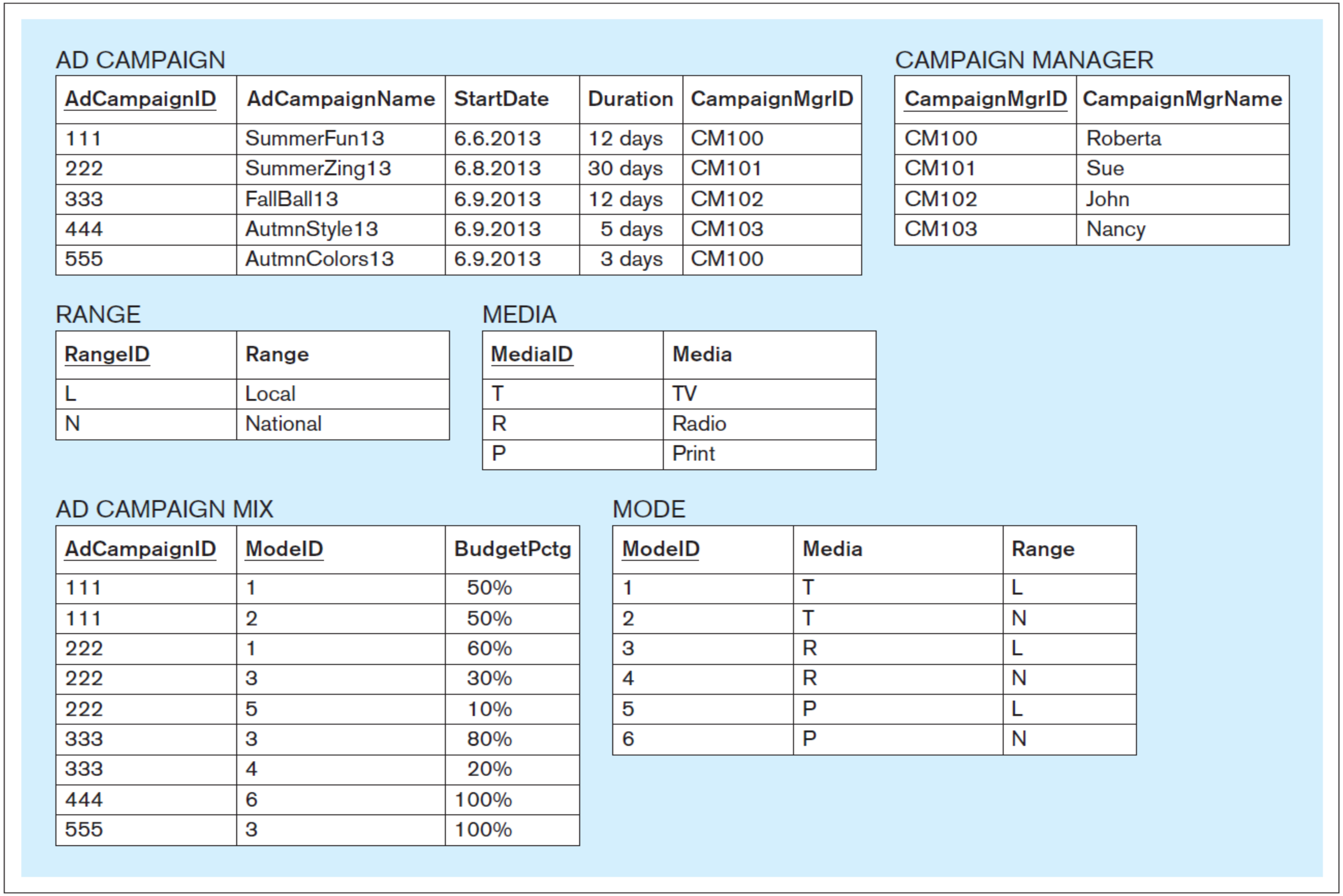
RANGE

RangeID
Range

MEDIA

MediaID
Media

Designer-added entities (tables) and keys example - mapped relations populated with data



AD CAMPAIGN

AdCampaignID	AdCampaignName	StartDate	Duration	CampaignMgrID
111	SummerFun13	6.6.2013	12 days	CM100
222	SummerZing13	6.8.2013	30 days	CM101
333	FallBall13	6.9.2013	12 days	CM102
444	AutmnStyle13	6.9.2013	5 days	CM103
555	AutmnColors13	6.9.2013	3 days	CM100

CAMPAIGN MANAGER

CampaignMgrID	CampaignMgrName
CM100	Roberta
CM101	Sue
CM102	John
CM103	Nancy

RANGE

RangeID	Range
L	Local
N	National

MEDIA

MediaID	Media
T	TV
R	Radio
P	Print

AD CAMPAIGN MIX

AdCampaignID	ModeID	BudgetPctg
111	1	50%
111	2	50%
222	1	60%
222	3	30%
222	5	10%
333	3	80%
333	4	20%
444	6	100%
555	3	100%

MODE

ModeID	Media	Range
1	T	L
2	T	N
3	R	L
4	R	N
5	P	L
6	P	N

ADDITIONAL STREAMLINING OF DATABASE CONTENT

Designer-added entities (tables) and keys

Augmenting databases with designer added tables and keys is not a default process that is to be undertaken in all circumstances.

Instead, augmenting databases with designer added tables and keys should be done judiciously, after analyzing pros and cons for each augmentation.

Conclusion

We have seen how Database Normalization can decrease redundancy, increase efficiency and reduce anomalies by implementing three of seven different levels of normalization called Normal Forms. The first three NFs are usually sufficient for most small to medium size applications.