Databases

Normalization II

Dr Konrad Dabrowski
konrad.dabrowski@durham.ac.uk

Online Office Hour:
Mondays 13:30–14:30
See Duo for the Zoom link

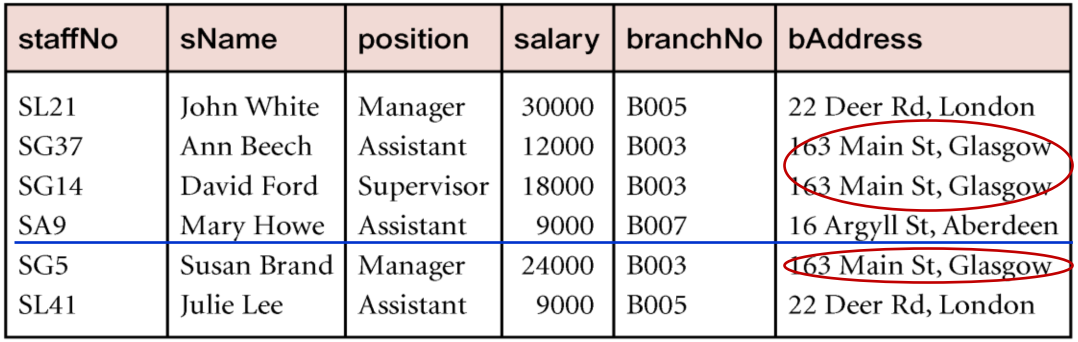
Construction of the Relational Data model

- Bottom-up approach: Normalization
  - start with the initial tables and attributes (from the ER model)
  - analyze the relationships among the attributes
  - re-design the tables and attributes in a "better" way:
    - decompose the tables into more tables (schema refinement)
    - ensure entity and referential integrity

- Purpose of normalization:
  - every relation represents a "real world" entity
  - single-valued columns
  - avoid redundancy (i.e. repetitions)
    - minimize the amount of space required
    - simplify maintenance of the database
  - data can be updated correctly
    - avoid update anomalies

Data update anomalies

StaffBranch



staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- Modification anomaly:
  - we want to change the address of branch B003
  - we must update it in many places (due to redundancy)
  - problem: if we do not update one of them ⟹ inconsistent data!
- Deletion anomaly:
  - we want to delete staff with staffNo SA9 from the database
  - this is the last staff member in branch B007
  - problem: we lose the details of this branch ⟹ incomplete data!
- Insertion anomaly:
  - we want to add a new branch, which has no staff yet ⟹ we must add Null into the attributes related to staff
  - staffNo is a primary key ⟹ violation of entity integrity!

Functional data dependencies

The fundamentals of normalization theory:

- Functional data dependency:
  - let A and B be two sets of attributes; we say that

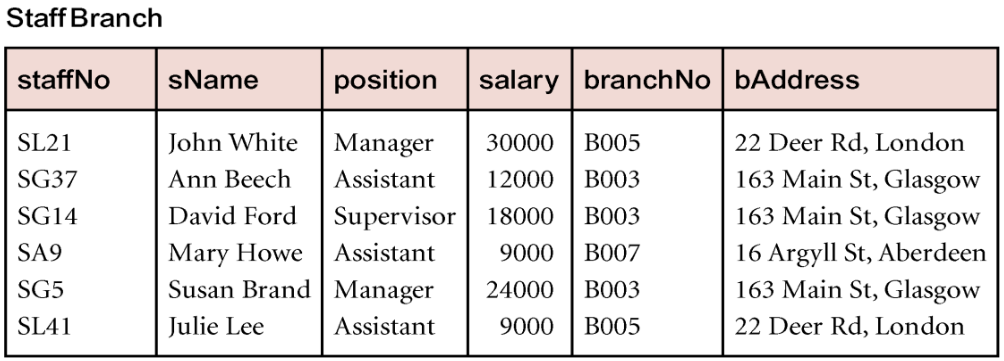
    "B is functionally dependent on A" (denoted $A \rightarrow B$)

    if each value of A is determines exactly one value of B

- In a functional data dependency ($A \rightarrow B$):
  - determinant: the set of all attributes on the left hand side (i.e. A)
  - dependent: the set of all attributes on the right hand side (i.e. B)

- By the definition of relational keys:
  - a candidate key is a minimal set of attributes, which functionally determine all attributes in a relation
  - among all candidate keys, we choose (any) one of them to serve as the primary key

Functional dependencies

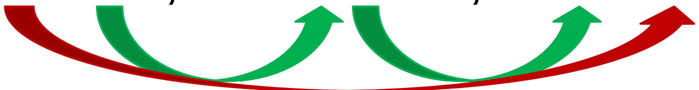


StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- Full functional dependency $A \rightarrow B$:
  - B is functionally dependent on A
  - B is not functionally dependent on any proper subset of A
  - example: staffNo → sName
- Partial functional dependency $A \rightarrow B$:
  - B is functionally dependent on A
  - B remains functionally dependent on at least one proper subset of A
  - example: staffNo, sName → branchNo

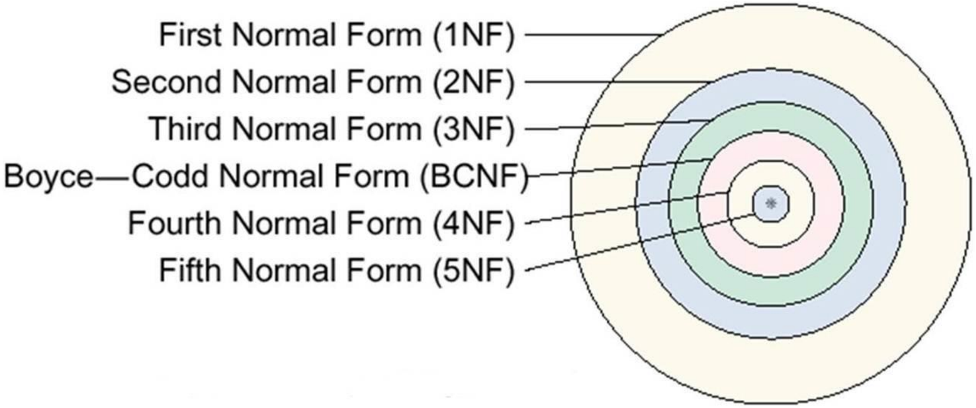
    (it suffices: staffNo → branchNo)
- Transitive functional dependency:
  - functional dependencies $A \rightarrow B$ and $B \rightarrow C$
  - then the functional dependency $A \rightarrow C$ is said to be transitive
  - example: staffNo, branchNo, bAddress



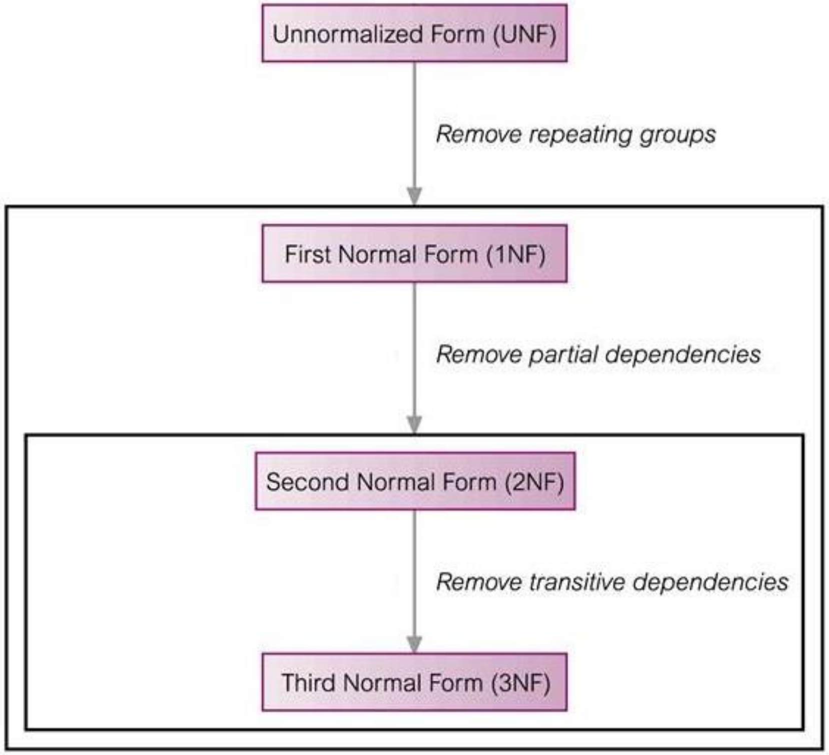
Normalization process

- Normalization is a multi-stage process
  - the result of each stage is called a Normal form
  - at each stage: check whether specific criteria are satisfied
    if not: re-organize the data

- We study the first 3 Normal forms
  - most important for practical applications



First Normal Form (1NF)
Second Normal Form (2NF)
Third Normal Form (3NF)
Boyce—Codd Normal Form (BCNF)
Fourth Normal Form (4NF)
Fifth Normal Form (5NF)



Unnormalized Form (UNF)

Remove repeating groups

First Normal Form (1NF)

Remove partial dependencies

Second Normal Form (2NF)

Remove transitive dependencies

Third Normal Form (3NF)

First Normal Form (1NF)

- Repeating group:
  - an attribute (or group of attributes) that occurs with multiple values for a single occurrence of the primary key
  - e.g. the attributes Item#, Qty, Part#, Desc

Note#	Packer	Name	Addr	Item#	Qty	Part#	Desc
300	JW	Bloggs	Perth	1 2 3	200 200 200	1234 2234 3334	Nuts Bolts Washer
301	SD	Smith	Durham	1 2	150 100	1234 3334	Nuts Washer

- A table is in un-normalised form (UNF):
  - when it contains one or more repeating groups
  - this does not conform with the definition of a relation
- A table is in the First Normal Form (1NF) if it has:
  - no repeating groups (every cell has one value)
  - no identical rows

First Normal Form (1NF)

How to bring a table in 1NF?

- one alternative would be:
  - repeat the appropriate columns horizontally

Note #	Packer	Name	Addr	Item#	Qty	Part#	Desc	Item#	Qty	Part#	Desc	Item#	Qty	Part#	Desc

Problem: a table must have a fixed number of columns

- we need a fixed (large) upper limit on the number of repetitions
- many of these new columns will be empty ⇒waste of space
- complicated querying: we need to search many columns to find e.g. the right Item#

- another alternative would be:
  - for every multi-valued attribute: one (long) string containing the whole list of items
  - the same problems: long strings, difficult querying

⇒ we need other solutions!

First Normal Form (1NF)

- First method: one-table solution
  - enter appropriate data in the empty columns (by repeating data)
  - i.e. fill in the blanks (also called “flattening the table”)

Packing note table

Note #	Packer	CoName	CoAddr	Item#	Qty	Part#	Desc
300	JW	Bloggs	Perth	1	200	1234	Nuts
300	JW	Bloggs	Perth	2	200	2234	Bolts
300	JW	Bloggs	Perth	3	200	3334	Washer
301	SD	Smith	Durham	1	150	1234	Nuts
301	SD	Smith	Durham	2	100	3334	Washer

- The resulting table is in 1NF
  - but still: we introduced a lot of redundancy (by repeating data)

First Normal Form (1NF)

- Second method: two-tables solution
  - place the repeating data in a separate relation
  - in the new relation place a copy of the original primary key
  - this key now becomes a foreign key (to refer to the original relation)
  - iterate until no repeated groups remain

Packing Note

Note#	Packer	CoName	CoAddr
300	JW	Bloggs	Perth
301	SD	Smith	Durham

Primary key

New table
(previously repeating data)

Packing Note Item

Note#	Item#	Qty	Part#	Desc
300	1	200	1234	Nuts
300	2	200	2234	Bolts
300	3	200	3334	Washer
301	1	150	1234	Nuts
301	2	100	3334	Washer

Foreign key

- The resulting tables are in 1NF
  - with much less redundancy than before

Second Normal Form (2NF)

- A table is in the Second Normal Form (2NF) if:
  - it is in 1NF and
  - there are no partial functional dependencies

    i.e. every non-key attribute is dependent on the whole primary key

- Non-key attributes:
  - all attributes that are not a part of the primary key

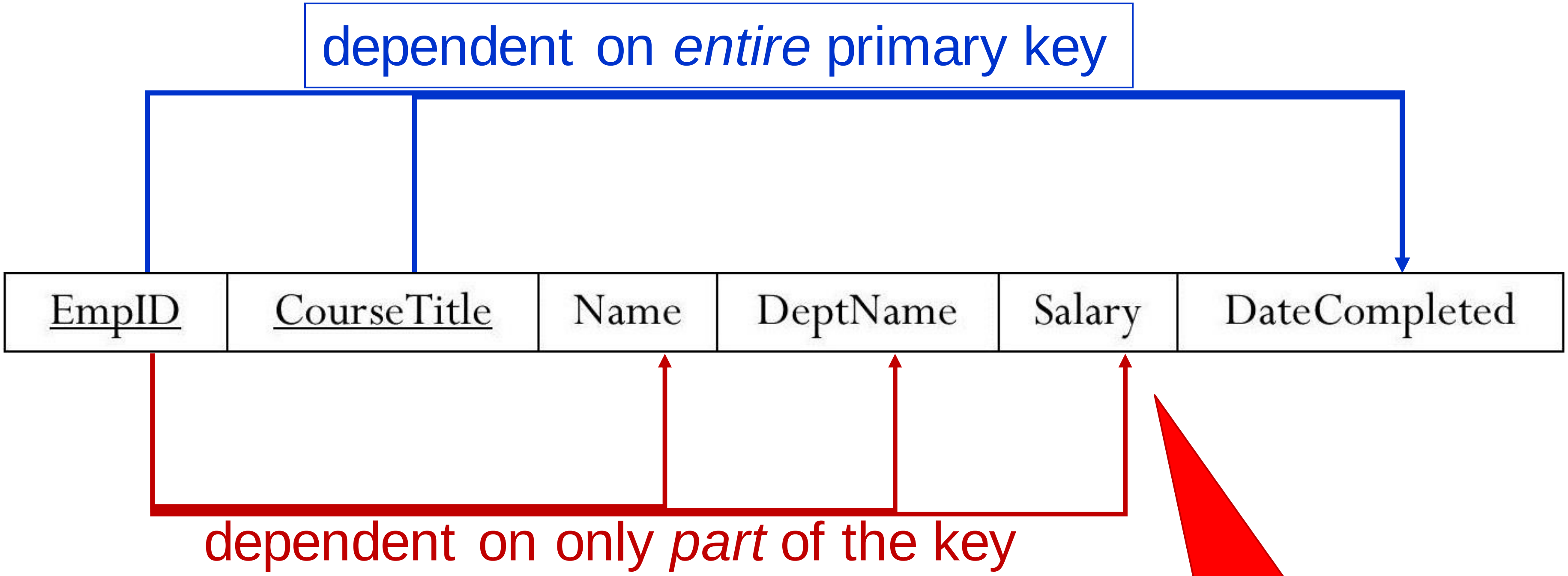
- 2NF applies to relations with composite keys

- When the primary key has only one attribute (simple key):
  - if the table is in 1NF ⇒ it is also in 2NF

- How to bring a table in 2NF:
  - remove the partially dependent attributes
  - place them in a new relation, along with the copy of their determinant

Second Normal Form (2NF)

- Example 1:



dependent on *entire* primary key

EmpID	CourseTitle	Name	DeptName	Salary	DateCompleted

dependent on only *part* of the key

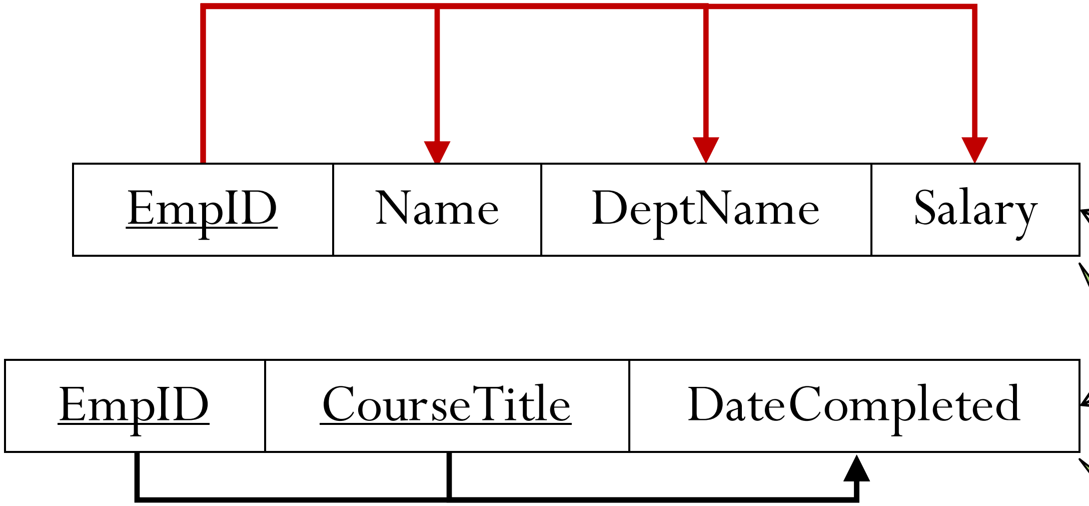
NOT in 2nd Normal Form !

EmpID, CourseTitle → DateCompleted

EmpID → Name, DeptName, Salary

Second Normal Form (2NF)

- Example 1 (converted in 2NF):
  - decompose into two separate relations



EmpID	Name	DeptName	Salary

EmpID	CourseTitle	DateCompleted

Both are full functional dependencies!

in 2nd Normal Form !

Second Normal Form (2NF)

- Example 2:
  - is this relation in 2NF? Why / why not?

NOT in 2nd Normal Form !

ENGINEER(Emp#, Name, Dept, Project#, ProjectName)

- Dependencies:
  - Emp# → Name
  - Emp# → Dept
  - Project# → ProjectName

all partial dependencies

Second Normal Form (2NF)

- Example 2:

NOT in 2nd Normal Form !

ENGINEER(Emp#, Name, Dept, Project#, ProjectName)

- Converted in 2NF: (correct?)

in 2nd Normal Form !

ENGINEER (Emp#, Name, Dept)

PROJECT (Project#, ProjectName)

  - Does this solve the problem? Any issues?

- Converted in 2NF: (correct!)

2NF and equivalent to the first one!

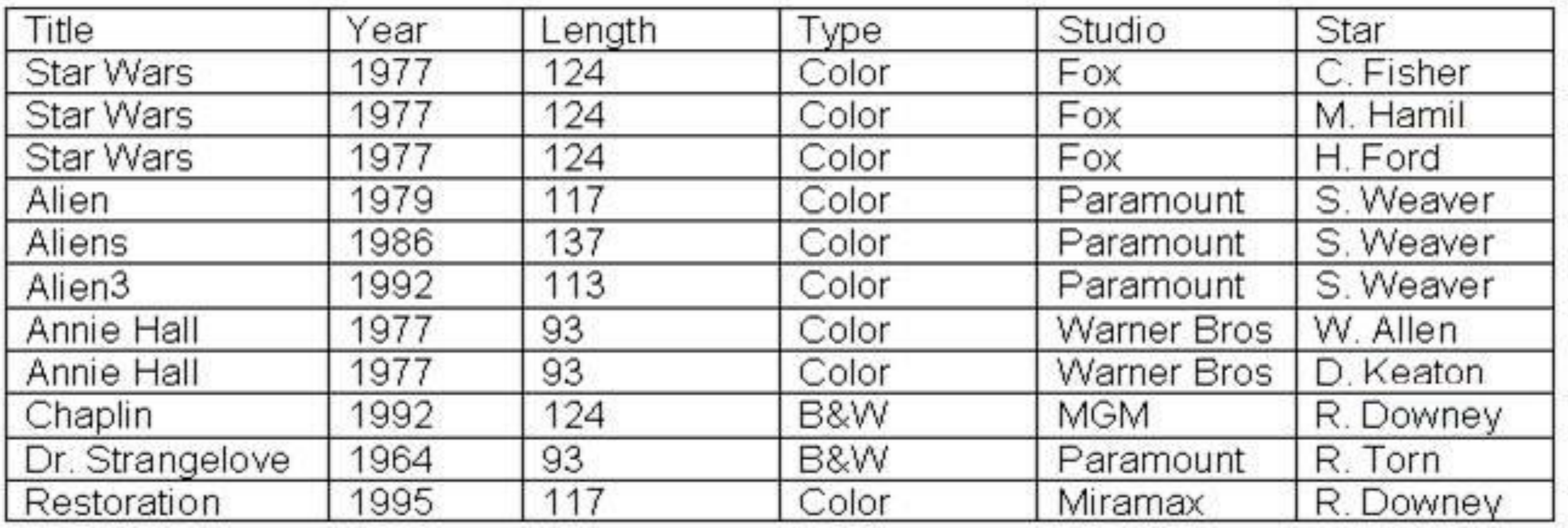
ENGINEER (Emp#, Name, Dept)

PROJECT (Project#, ProjectName)

ENGINEER_PROJECT (Emp#, Project#)

Second Normal Form (2NF)

- Example 3: Information about movies including their main stars
  - one movie can have many stars ⟹ primary key is (Title, Star)



Title	Year	Length	Type	Studio	Star
Star Wars	1977	124	Color	Fox	C. Fisher
Star Wars	1977	124	Color	Fox	M. Hamil
Star Wars	1977	124	Color	Fox	H. Ford
Alien	1979	117	Color	Paramount	S. Weaver
Aliens	1986	137	Color	Paramount	S. Weaver
Alien3	1992	113	Color	Paramount	S. Weaver
Annie Hall	1977	93	Color	Warner Bros	W. Allen
Annie Hall	1977	93	Color	Warner Bros	D. Keaton
Chaplin	1992	124	B&W	MGM	R. Downey
Dr. Strangelove	1964	93	B&W	Paramount	R. Torn
Restoration	1995	117	Color	Miramax	R. Downey

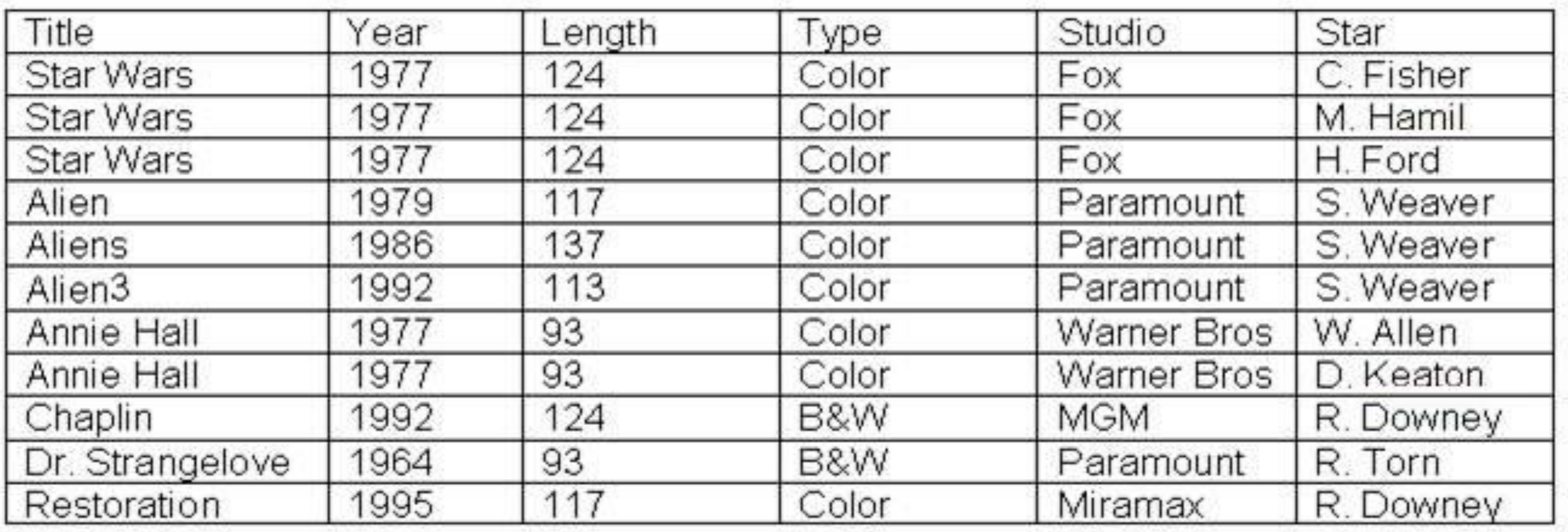
NOT in 2nd Normal Form !

- Is it in 1NF? YES
- Functional dependencies:
  - Title → Year
  - Title → Length
  - Title → Type
  - Title → Studio

all partial dependencies

Second Normal Form (2NF)

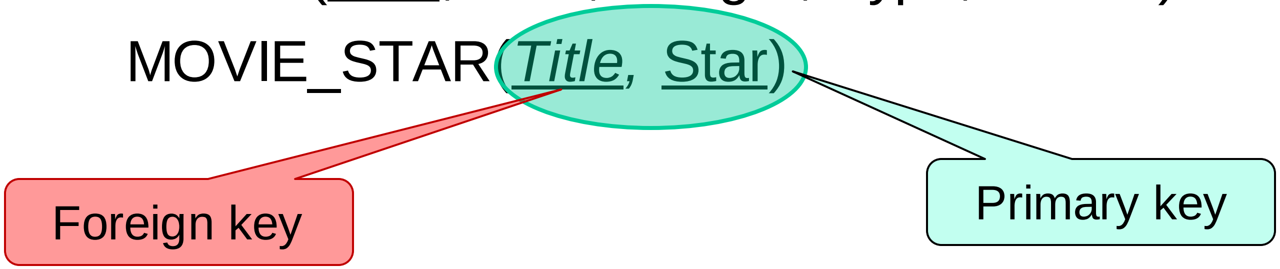
- Example 3: Information about movies including their main stars
  - one movie can have many stars ⟹ primary key is (Title, Star)



Title	Year	Length	Type	Studio	Star
Star Wars	1977	124	Color	Fox	C. Fisher
Star Wars	1977	124	Color	Fox	M. Hamil
Star Wars	1977	124	Color	Fox	H. Ford
Alien	1979	117	Color	Paramount	S. Weaver
Aliens	1986	137	Color	Paramount	S. Weaver
Alien3	1992	113	Color	Paramount	S. Weaver
Annie Hall	1977	93	Color	Warner Bros	W. Allen
Annie Hall	1977	93	Color	Warner Bros	D. Keaton
Chaplin	1992	124	B&W	MGM	R. Downey
Dr. Strangelove	1964	93	B&W	Paramount	R. Torn
Restoration	1995	117	Color	Miramax	R. Downey

- Converted in 2NF:

MOVIE(Title, Year, Length, Type, Studio)



MOVIE_STAR(*Title*, Star)

Foreign key

Primary key

Third Normal Form (3NF)

- A table is in the Third Normal Form (3NF) if:
  - it is in 2NF and
  - there are no transitive functional dependencies

    i.e. no non-key attribute is transitively dependent on the primary key

- In other words, in 3NF:
  - all attributes (which are not part of the primary key) are functionally dependent on the key, the whole key, and nothing but the key

- How to bring a table in 3NF:
  - remove the transitively dependent attributes
  - place them in a new relation
  - take the attributes of their determinant as the primary key in the new table

Third Normal Form (3NF)

- In one of the previous examples:

PackingNote (<u>Note#,</u> Packer, CompanyName, CompanyAddr)

- CompanyAddr is transitively dependent on <u>Note#</u> via CompanyName

⇒ after removing this transitive dependency:

PackingNote (<u>Note#,</u> Packer, CompanyName)

Company (<u>CompanyName</u>, CompanyAddr)

in 3NF!

Foreign key

PackingItem (<u>Note#,</u> <u>Item#,</u> Qty, Part#, Desc)

- Desc is transitively dependent on <u>Note#</u> and <u>Item#</u> via Part#

⇒ after removing this transitive dependency:

PackingItem (<u>Note#,</u> <u>Item#</u>, Qty, Part#)

Part (<u>Part#</u>, Desc)

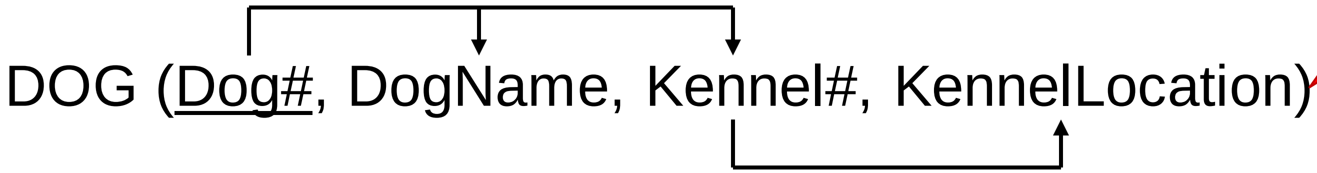
in 3NF!

Foreign key

Third Normal Form (3NF)

- Example 4:
  - is this relation in 3NF? Why / why not?

NOT in 3NF !


DOG (Dog#, DogName, Kennel#, KennelLocation)

- Dependencies:
  - Dog# → DogName
  - Dog# → Kennel#
  - Kennel# → KennelLocation

transitive dependency

- Converted to 3NF:

Foreign key

DOG (Dog#, DogName, Kennel#)

KENNEL (Kennel#, KennelLocation)

Third Normal Form (3NF)

NOT in 3NF !

- Example 5:

VIN = Vehicle Id. Number
NIN = National Insurance Number

VIN	Make	Model	Year	NIN	Owner
111abc	Toyota	Corolla	1988	111223333	Joe Smith
223ahv	Ford	Windstar	1998	222334444	Bill Gates
332amz	GM	GMC	1995	333445555	Tom Green
876grd	Subaru	Outback	2000	987654321	Bob Jones

- This relation is in 1NF
  - no composite primary key ⟹ also in 2NF

- Dependencies:
  - VIN → Make
  - VIN → Model
  - VIN → Year
  - VIN → NIN
  - NIN → Owner

transitive dependency

- Converted to 3NF:

VEHICLE (VIN, Make, Model, Year, NIN)
OWNER (NIN, OwnerName)

Foreign key

Third Normal Form (3NF)

NOT in 3NF !

- Example 6:

Cust_ID	Name	SalesPerson	ShopRegion
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West

- This relation is in 1NF
  - no composite primary key ⟹ also in 2NF

- Dependencies:

transitive dependency

  - Cust_ID → Name
  - Cust_ID → SalesPerson
  - Cust_ID → ShopRegion
  - SalesPerson → ShopRegion

- Converted to 3NF:

Foreign key

SALES (Cust_ID, Name, SalesPerson)

SALESPERSON (SalesPerson, ShopRegion)

Third Normal Form (3NF)

NOT in 3NF !

- Example 7:

ShipmentNum	Origin	Destination	Distance
409	Seattle	Denver	1,537
618	Chicago	Dallas	1,058
723	Boston	Atlanta	1,214
824	Denver	Los Angeles	1,150
629	Minneapolis	St. Louis	587

- This relation is in 1NF
  - no composite primary key ⇒ also in 2NF

transitive dependency

- Dependencies:
  - ShipmentNum → Origin, Destination, Distance
  - Origin, Destination → Distance

- Converted to 3NF:

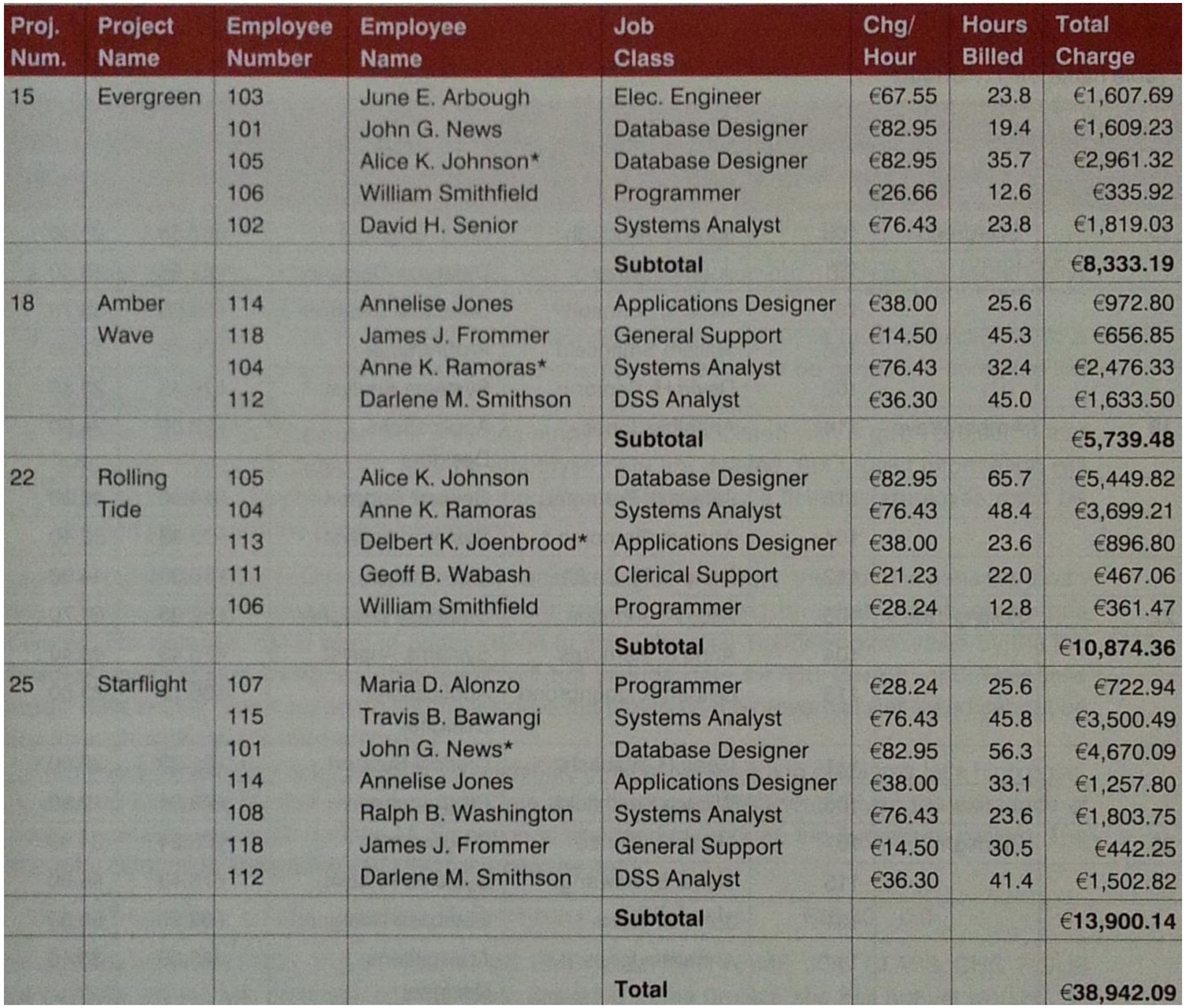
Foreign key

SHIPTO (ShipmentNum, Origin, Destination)

DISTANCE (Origin, Destination, Distance)

A full normalization example

- Un-normalized data (sample report layout):
  - many repeating groups
  - not in 1NF
- The “Subtotal” items:
  - derived attributes (can be computed by the other attributes)
  - they don’t need to be in the database
- We convert it to 1NF:
  - e.g. by flattening the table

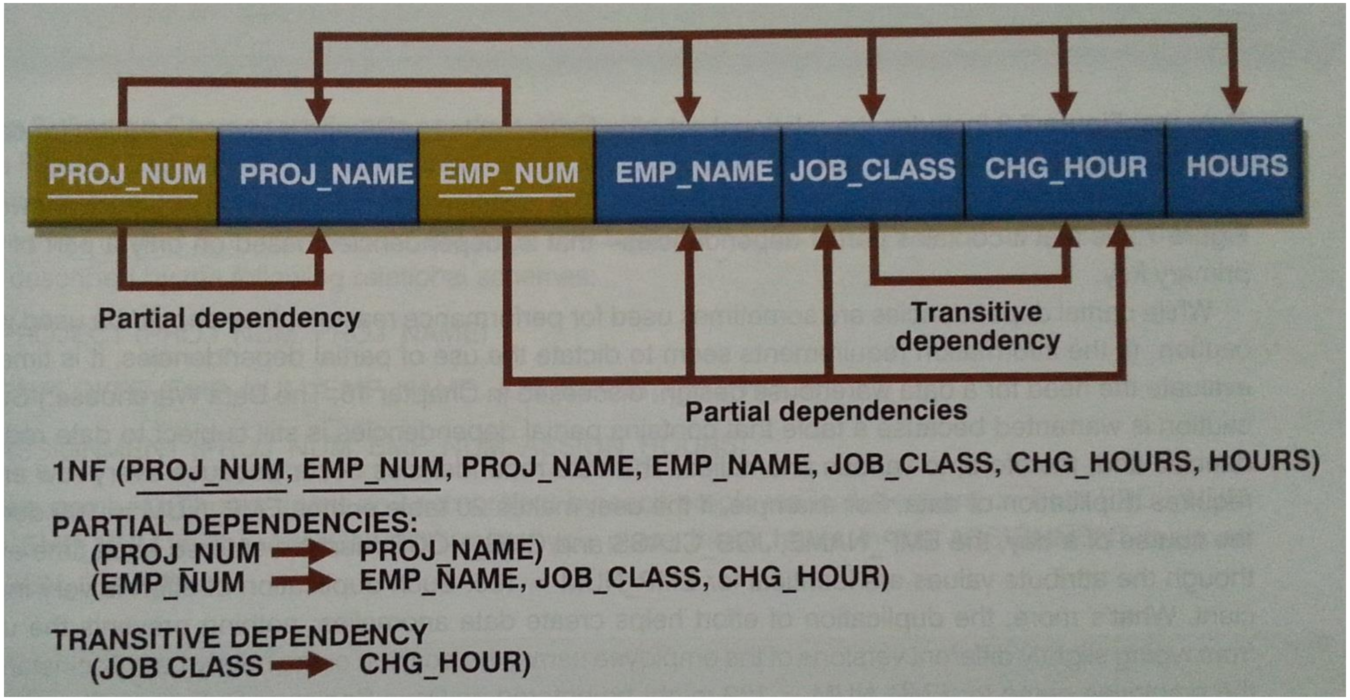


Proj. Num.	Project Name	Employee Number	Employee Name	Job Class	Chg/ Hour	Hours Billed	Total Charge
15	Evergreen	103	June E. Arbough	Elec. Engineer	€67.55	23.8	€1,607.69
		101	John G. News	Database Designer	€82.95	19.4	€1,609.23
		105	Alice K. Johnson*	Database Designer	€82.95	35.7	€2,961.32
		106	William Smithfield	Programmer	€26.66	12.6	€335.92
		102	David H. Senior	Systems Analyst	€76.43	23.8	€1,819.03
				Subtotal			**€8,333.19**
18	Amber Wave	114	Annelise Jones	Applications Designer	€38.00	25.6	€972.80
		118	James J. Frommer	General Support	€14.50	45.3	€656.85
		104	Anne K. Ramoras*	Systems Analyst	€76.43	32.4	€2,476.33
		112	Darlene M. Smithson	DSS Analyst	€36.30	45.0	€1,633.50
				Subtotal			**€5,739.48**
22	Rolling Tide	105	Alice K. Johnson	Database Designer	€82.95	65.7	€5,449.82
		104	Anne K. Ramoras	Systems Analyst	€76.43	48.4	€3,699.21
		113	Delbert K. Joenbrood*	Applications Designer	€38.00	23.6	€896.80
		111	Geoff B. Wabash	Clerical Support	€21.23	22.0	€467.06
		106	William Smithfield	Programmer	€28.24	12.8	€361.47
				Subtotal			**€10,874.36**
25	Starflight	107	Maria D. Alonzo	Programmer	€28.24	25.6	€722.94
		115	Travis B. Bawangi	Systems Analyst	€76.43	45.8	€3,500.49
		101	John G. News*	Database Designer	€82.95	56.3	€4,670.09
		114	Annelise Jones	Applications Designer	€38.00	33.1	€1,257.80
		108	Ralph B. Washington	Systems Analyst	€76.43	23.6	€1,803.75
		118	James J. Frommer	General Support	€14.50	30.5	€442.25
		112	Darlene M. Smithson	DSS Analyst	€36.30	41.4	€1,502.82
				Subtotal			**€13,900.14**
				Total			**€38,942.09**

* For the full example, see: “Database Systems: Design, Implementation & Management”, by P. Rob, C. Coronel, K. Crockett

A full normalization example

- We find an adequate primary key in the resulting 1NF table:
  - composite key (Proj_Num, Emp_Num)
  - this can be found by computing the functional dependency closure (or just by intuitive observations from the data)
- We identify all functional dependencies
  - we depict them in a dependency diagram



1NF (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOURS, HOURS)

PARTIAL DEPENDENCIES:
(PROJ_NUM → PROJ_NAME)
(EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR)

TRANSITIVE DEPENDENCY
(JOB CLASS → CHG_HOUR)

A full normalization example

- To convert the table to 2NF:
  - write each primary key component on a separate line:

    PROJ_NUM

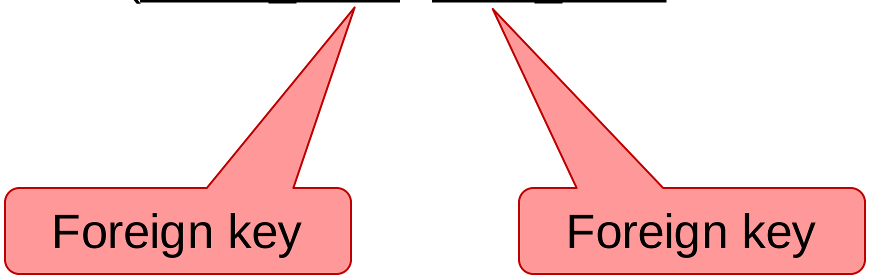
    EMP_NUM

    PROJ_NUM, EMP_NUM
  - each of these components will become the key in a new table
  - assign corresponding dependent attributes

    PROJECT (PROJ_NUM, PROJ_NAME)

    EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

    ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN HOURS)



Foreign key

Foreign key

  - from all these new tables, we now depict all functional dependencies in another dependency diagram

A full normalization example

- To convert the table to 2NF:
  - from all these new tables, we now depict all functional dependencies in another dependency diagram
  - There are no partial dependencies now ⇒ table in 2NF

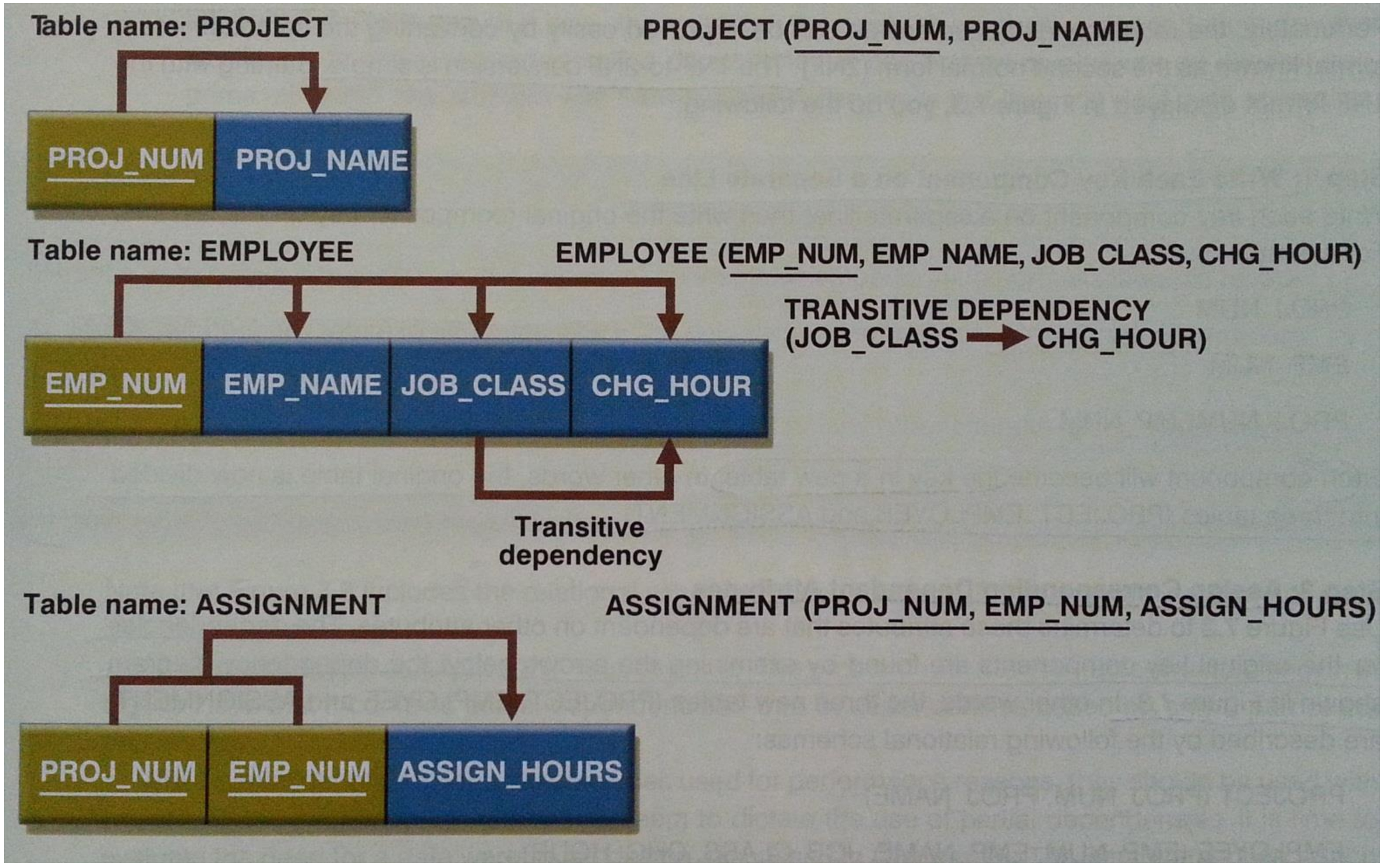


Table name: PROJECT

PROJECT (PROJ_NUM, PROJ_NAME)

Table name: EMPLOYEE

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

TRANSITIVE DEPENDENCY (JOB_CLASS → CHG_HOUR)

Transitive dependency

Table name: ASSIGNMENT

ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

A full normalization example

- To convert the table to 3NF:
  - for every transitive dependency, write its determinant as a primary key in a new table; here the determinant is:

    JOB_CLASS

  - identify the dependent attributes (which are dependent on each determinant); here we have:

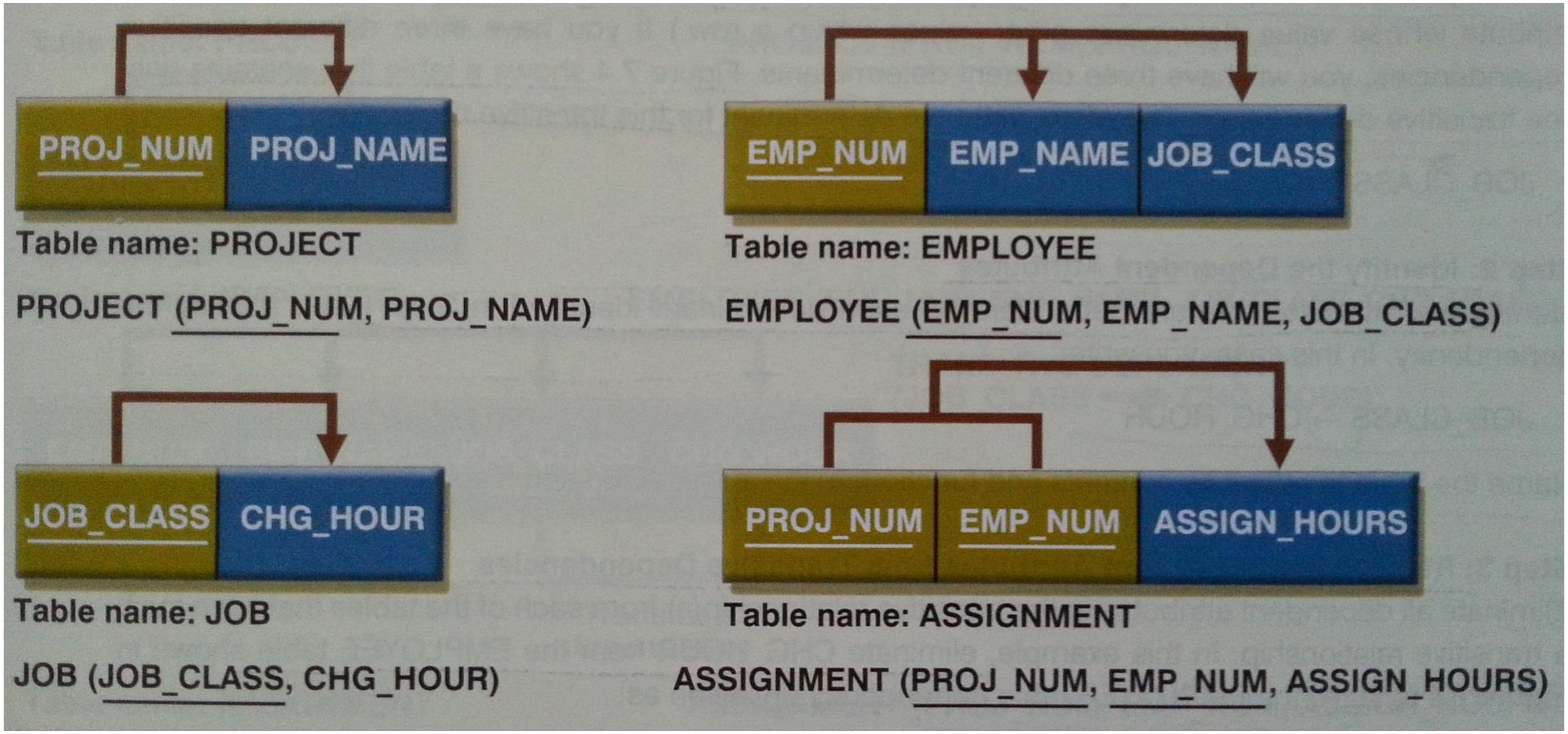
    JOB_CLASS → CHG_HOUR

  - remove the dependent attributes from transitive dependencies; here we remove CHG_HOUR from Employee

  - There are no transitive dependencies now ⇒ table in 3NF

A full normalization example

- The dependency diagram of 3NF:



PROJ_NUM PROJ_NAME

Table name: PROJECT

PROJECT (PROJ_NUM, PROJ_NAME)

EMP_NUM EMP_NAME JOB_CLASS

Table name: EMPLOYEE

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

JOB_CLASS CHG_HOUR

Table name: JOB

JOB (JOB_CLASS, CHG_HOUR)

PROJ_NUM EMP_NUM ASSIGN_HOURS

Table name: ASSIGNMENT

ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)