Relational model

COSC430—Advanced Databases
David Eyers

Learning objectives and references

- You should be able to:
  - define the elements of the relational model;
  - determine functional dependencies (FDs) for example databases;
  - derive functional dependencies by applying inference axioms;
  - apply closures of FDs' attributes to explore a relation's structure;
  - understand and use the relational data operators and their notation;
  - define different types of database normalisation;
  - determine the normal form that a relation is in (1NF,2NF,3NF,BCNF);
  - decompose relations into a given normal form.
- Elmasri chapters of relevance: 6th ed., ch3; ch6; ch14

Example relational database

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	9-Jan-1965	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	8-Dec-1955	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-Jul-1968	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-Jun-1941	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-Sep-1962	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-Jul-1972	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-Mar-1969	980 Dallas, Houston, YX	M	25000	987654321	4
James	E	Borg	888665555	10-Nov-1937	450 Stone, Houston, TX	M	55000	NULL	1

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerisation	10	Stafford	4
Reorganisation	20	Houston	1
NewBenefits	30	Stafford	4

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-May-1988
Administration	4	987654321	1-Jan-1995
Headquarters	1	888665555	19-Jun-1981
Dummies	0	111100000	31-Dec-2004

DEPT_LOCATION

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	5-Apr-1986	Daughter
333445555	Theodore	M	25-Oct-1983	Son
333445555	Joy	F	3-May-1958	Spouse
987654321	Abner	M	28-Feb-1942	Spouse
123456789	Michael	M	4-Jan-1988	Son
123456789	Alice	F	30-Dec-1988	Daughter
123456789	Elizabeth	F	5-May-1967	Spouse

Relational schema of COMPANY database

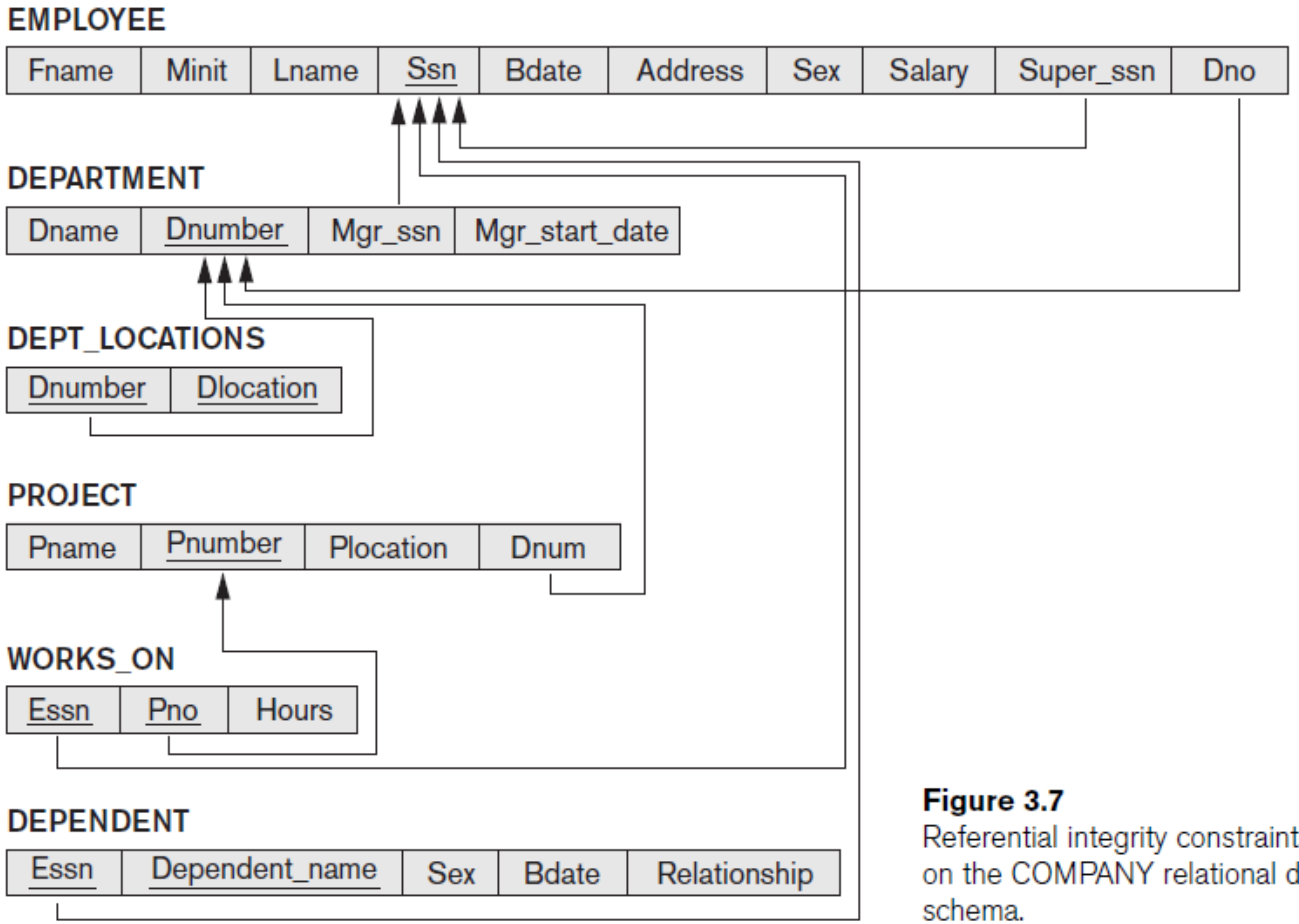


Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

The Relational data model

- Invented by E.F. Codd, IBM Research in 1970
  - E.F. Codd, "A Relational Model for Large Shared Data Banks", *Communications of the ACM*, 13:6, June 1970
- Has formal basis in mathematics
  - Set theory
  - First order predicate logic
- The dominant model for database systems
  - Oracle, 1979
  - IBM DB2, 1983
  - Microsoft SQL Server, 1989
  - ... and all the open source offerings

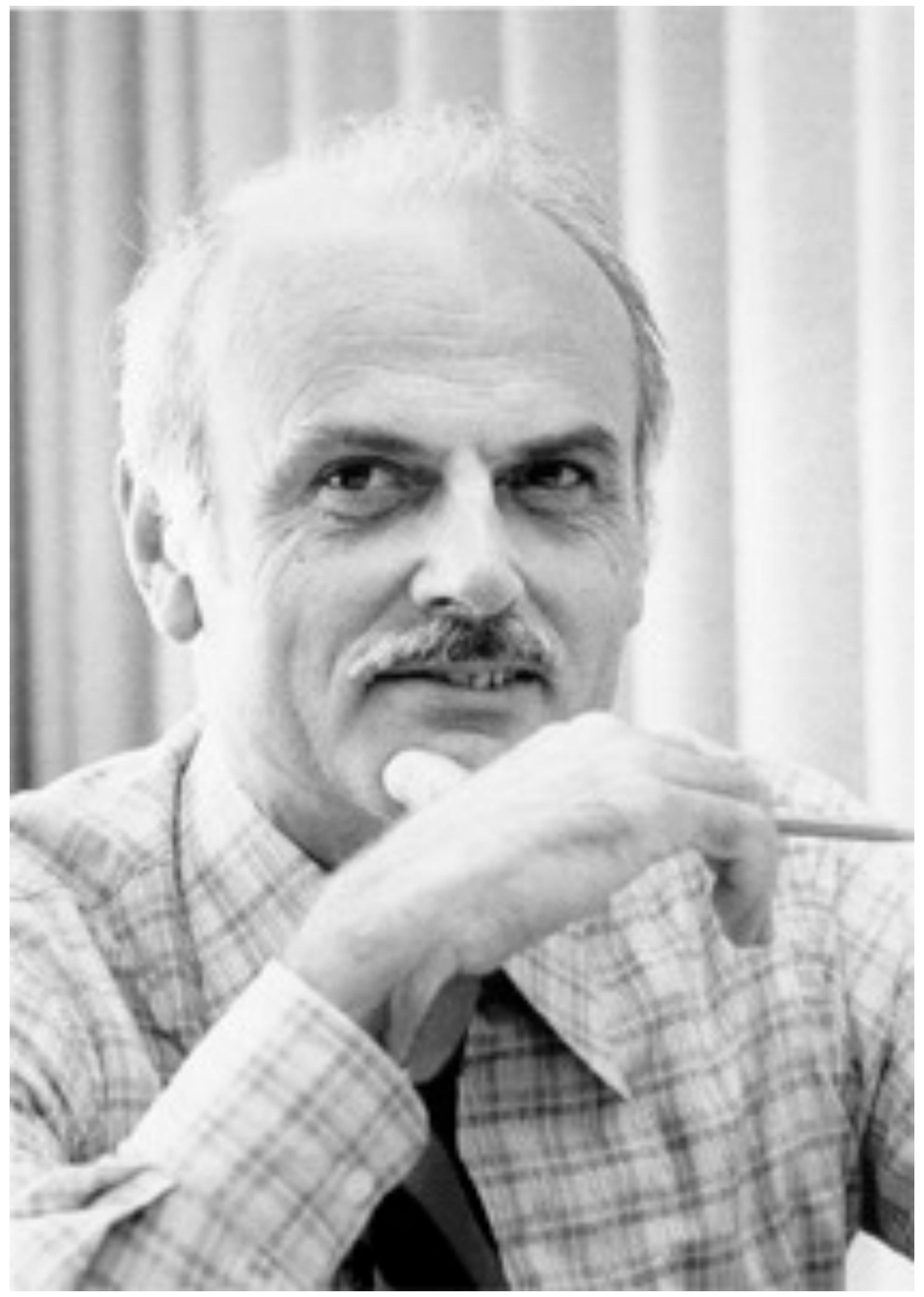


Image copyright IBM

Relational data model components

- The three core components in the relational model:

- Objects or **relations**—the structure of data organisation
- **Integrity constraints**—enforcing constraints and rules
- **Operators**—manipulation of data

- We first examine **relations**

Fundamental terms

- **relation**—instance of a schema
- **attribute**—one element within a tuple
- **domain**—set from which attribute values can come
- **tuple**—mapping from schema into attributes' domain
  - cardinality—number of tuples
  - degree—number elements in tuple
- **key**—means to identify tuple

SNUM	SNAME	STATUS	CITY

Formalisation of relations

- A **relation scheme** *R* denoted $R(A_1, A_2, \ldots, A_n)$ is made up of:
  - The name of the relation *R*
  - A set of **attributes** $\{A_1, A_2 .. A_n\}$.
- Corresponding to each attribute name A_i is a set D_i, $1 \leq i \leq n$, called the **domain** of A_i, sometimes denoted by $dom(A_i)$.
- Let $\boldsymbol{D} = D_1 \cup D_2 \cup \ldots\ldots. \cup D_n$
- A **relation** *r* on relation scheme *R* is a finite set of mappings $\{t_1, t_2, \ldots.. t_p\}$ from *R* to $\boldsymbol{D}$ with the restriction that for each mapping $t \in r$, $t(A_i)$ must be in D_i, $1 \leq i \leq n$.
- The mappings are called ***tuples***.

(from David Maier, The theory of relational databases, Pitman, 1983)

Relational data model components

- The three core components in the relational model:

- Objects or **relations**—the structure of data organisation
- **Integrity constraints**—enforcing constraints and rules
- **Operators**—manipulation of data

- Now let's look at **integrity constraints**

Constraints are everywhere

- Situations which lead to data restrictions or constraints in the modelled world:
  - *Every student has a unique student ID*
  - *You can't be in two places at the same time*
  - *A truck driver can drive for 11 hours max., and work for a 14 hours max. in a day, before having to take 10 hours off duty or in the sleeper*
  - *Maximum room capacity 24*
  - *Speed limit 30*

Constraints in the relational model

- Domain constraints
- Key constraints
  - **superkey**—set of attributes that can identify a tuple
  - **candidate key**—minimal super key
  - **primary key**—chosen candidate key
- Entity Integrity constraint
  - No primary key values can be NULL
- Referential Integrity constraint
  - Foreign keys interlink data in a valid way

Constraints are everywhere

- Situations which lead to data restrictions or constraints in the modelled world:
  - *Every student has a unique student ID*
  - *You can't be in two places at the same time*
  - *A truck driver can drive for 11 hours max., and work for a 14 hours max. in a day, before having to take 10 hours off duty or in the sleeper.*
  - *Maximum room capacity 24*
  - *Speed limit 30*
- Some can be expressed by a **functional dependency**, e.g.,
  - {person, time, date} → place   → = "determines"

Functional dependencies

*The single most important concept in **relational schema design** is that of a functional dependency*—Elmasri, p497

- A **functional dependency** (dependence) is a many-to-one relationship from one set of attributes to another
- Given a relation *R*, attribute Y of *R* is functionally dependent on attribute X of *R* if and only if:
  - in every possible legal value of *R* each X-value has associated with it precisely one Y-value

Healthy Harbour Watchers

- … is a community group that collects data about the quality of our foreshore environments

1. Pulling Point
2. Back Beach, Port Chalmers
3. Mussel Bay
4. Ravensbourne Boat Club
5. Leith River Mouth

FDs exist in every relation

Sample (S#, PLACE, DATE, TEAM, LNAME, LPH, HT, WT, WPH, …) where:

- S#, Sample number
- PLACE, Place where sample is taken
- DATE, Date when sample taken
- TEAM, Which team collected the sample
- LNAME, Team leader's name
- LPH, Team leader's contact phone
- HT, High tide
- WT, water temperature
- WPH, water pH value …

- List some functional dependencies in relation Sample, stating any assumptions you make. Identify a candidate key

Sample data values

Sample #	Pla	Date	Team	LName	LPh	WT	…
81	PP	d1	t1	Alex	021 123	10.3	
82	PP	d1	t1	Alex	021 123	10.3	
83	BB	d1	t1	Alex	021 123	10.9	
84	MB	d1	t1	Alex	021 123	11.0	
81	PP	d2	t1	Alex	021 123	9.9	
82	PP	d2	t1	Alex	021 123	9.9	
83	BB	d2	t2	Kathy	022 246	10.1	
84	MB	d2	t2	Kathy	022 246	10.4	

Inference axioms

- For a relation *R* there is a family of FDs, *F*, that *R* satisfies
  - Finding *F* requires some **semantic knowledge** of *R*
- Knowing some members of *F*, it is often possible to infer other members of *F*
- An **inference axiom** is a rule that states if a relation satisfies certain FDs then it must satisfy certain other FDs

For example, a transitive dependency

- If we have a relation

Staff ID	Full name	Address
12345	Jo Smith	99 High Street
6789	Ken Jones	99 Leith Street

- and: Staff_ID → Full name
- and: Full name → Address
- then we can infer: Staff_ID → Address

- if a relation satisfies certain FDs, it must satisfy certain other FDs

Maier's list of inference axioms

F1 Reflexivity
$X \rightarrow X$

F2 Augmentation
$X \rightarrow Y$ implies $XZ \rightarrow Y$
(Elmasri $X \rightarrow Y$ implies $XZ \rightarrow YZ$)

F3 Additivity *(Union—Elmasri)*
$X \rightarrow Y$ and $X \rightarrow Z$ implies $X \rightarrow YZ$

F4 Projectivity
(Decomposition—Elmasri)
$X \rightarrow YZ$ implies $X \rightarrow Y$

F5 Transitivity
$X \rightarrow Y$ and $Y \rightarrow Z$ implies $X \rightarrow Z$

F6 Pseudotransitivity
$X \rightarrow Y$ and $YW \rightarrow Z$ imply $XW \rightarrow Z$

See—David Maier, The theory of relational databases, Pitman, 1983

Armstrong's axioms

- Armstrong's actual axioms are:
  - Reflexivity (F1)
  - Augmentation (F2)
  - Pseudotransitivity (F6)
- … and the others can be derived from them

- Elmasri (p529) uses the term **inference rules** and defines Armstrong's rules as
  - reflexivity
  - augmentation
  - transitivity

Armstrong's axioms are

Complete

- Given a set *F* of FDs, all the FDs implied by *F* can be derived using Armstrong's axioms

Sound

- Given a set *F* of FDs, no FDs not implied by *F* will be derived using Armstrong's axioms

The closure of a set of **dependencies**

- The closure of *F*
- The set of all FDs implied by a given set *F* is called the closure of *F*, denoted by $\mathbf{F^+}$
- $\mathbf{F^+}$ is the smallest set containing *F* such that Armstrong's axioms cannot be applied to the set to yield an FD not in the set

For example

Given $R(A, B, C)$

and $F = \{A \rightarrow B, B \rightarrow C\}$

$\mathbf{F^+} =$ { A → A, B → B, C → C, AB → AB,

AC → AC, BC → BC, ABC → ABC,

AB → A,………………

A → B, AB → B, AC → B, ABC → B,

B → C, AB → C, BC → C, ABC → C,

B → BC, A → ABC,…………………………}

The closure of a set of **attributes**

- Given a set of FDs, F and a set of attributes X
- The closure of X, denoted by $\mathbf{X^+}$ is the maximal set of attributes determined by X, within the closure of the set of FDs, $\mathbf{F^+}$

For example

Given R(A, B, C) and $F = \{A \rightarrow B, B \rightarrow C\}$

F⁺ = { **A → A**, B → B, C → C, AB → AB, AC → AC, BC → BC, ABC → ABC, AB → A,........ **A → AB**.........
A → B, AB → B, AC → B, ABC → B, **A → C**, B → C,
AB → C, BC → C, ABC → C, B → BC,
A → AC,.................. **A → ABC**,.........}

A^+ = ABC

B^+ = BC

BC^+ = BC

Minimal sets of dependencies

- For every set of FDs E, there is a covering set F with the following properties
  - every RHS is a single attribute
  - every LHS is irreducible—no attribute is redundant
  - no FD is F is redundant
- This minimal set F is referred to as a **minimal cover** (or an *irreducible set*)
- It is a set of dependencies in a *standard form* and has *no redundancies*

Minimal cover—example

R (A B C D J)

$F = \{ A \rightarrow B, AB \rightarrow D, C \rightarrow AD, C \rightarrow J \}$

$F = \{ A \rightarrow B, AB \rightarrow D, C \rightarrow A, C \rightarrow D, C \rightarrow J \}$

$F = \{ A \rightarrow B, AB \rightarrow D, C \rightarrow A, C \rightarrow D, C \rightarrow J \}$

$F = \{ A \rightarrow B, A \rightarrow D, C \rightarrow A, C \rightarrow D, C \rightarrow J \}$

$F = \{ A \rightarrow B, A \rightarrow D, C \rightarrow A, C \rightarrow D, C \rightarrow J \}$

$F = \{ A \rightarrow B, A \rightarrow D, C \rightarrow A, C \rightarrow J \}$

Testing membership in F+

Given a relation R (A B C)

and a set of FDs F = { AB → C, C → B }

does F imply AC → AB ?

Compute AC^+ , the closure of AC under F

Then, if AB is a subset of AC^+ **YES** else **NO**

In general, to determine if a set of FDs, F logically implies X → Y

Compute X^+ , the closure of X, then, if Y is a subset of X^+ **YES** else **NO**

The Closure Algorithm

(*i.e.*, the closure of a set of **attributes**)

Given: a set of FDs F and a set of attributes X

```
Begin
        OLD := { }; NEW := {X};
        while OLD <> NEW do
                OLD := NEW;
                for every FD W → Z in F do
                        if W ⊆ NEW then
                                NEW := NEW ∪ Z
                end
        end
end.
```

Example 1

Given: R (ABCDEJ)

F = { A → D, AB → E, BJ → E, CD → J, E → C }

Does F imply AE → BJ ?

Compute $(AE)^+$

OLD := { } NEW := {AE}

OLD := {AE} NEW := {AE D C}

OLD := {ACDE} NEW := {ACDE J}

OLD := {ACDEJ} NEW := {ACDEJ}

$(AE)^+$ = {ACDEJ} So F does not imply AE → BJ

Example 2

Given: R (ABCD)

$F = \{ A \rightarrow B, C \rightarrow B, A \rightarrow D, D \rightarrow C, B \rightarrow A \}$

Is $A \rightarrow B$ redundant?

Compute A^+ ignoring $A \rightarrow B$

$F = \{ A \rightarrow B, C \rightarrow B, A \rightarrow D, D \rightarrow C, B \rightarrow A \}$

FDs and keys

- A **superkey** X of relation *R* is a set of attributes that uniquely identifies a tuple:
  - **1:** $X \rightarrow A_1, A_2 \ldots A_n$ is in F^+
    - so attribute set X can functionally determine every attribute in R
- X is a **candidate key** if condition 1 holds, and also:
  - **2:** For no proper subset $Y \subset X$ is $Y \rightarrow A_1, A_2 \ldots A_n$ in F^+
    - so X is minimal: there is no subset of X that is a superkey
- Note that the **set of all attributes** must be either a candidate key or a superkey

Example

Given: R (A, B, C, D, E, G)

$F = \{A \rightarrow B, BC \rightarrow DE, AE \rightarrow G\}$

Is A a candidate key? Compute A^+

Is AC a candidate key? Compute AC^+

Is ABC a candidate key? Compute ABC^+

Then, if necessary, check if A or AC or ABC is a superkey

Dependency preservation and the projection of a set of dependencies

- Let *F* be a set of dependencies for *R*
- Projection of *F* onto a set of attributes Z, denoted $\Pi_{(Z)}(F)$, is the set of dependencies, $X \rightarrow Y$, in F^+ such that XY is a subset of Z. For example:

R (A B C) F = { $A \rightarrow B$, $B \rightarrow C$ }

R1 (A B)

$H^+ = \{A \rightarrow A, AB \rightarrow A, B \rightarrow B,$
$AB \rightarrow B, AB \rightarrow AB, A \rightarrow B,$
$A \rightarrow AB\}$

R2 (A C)

$K^+ = \{......\}$

Example—post codes

Street	Town	Post code
High St	Dunedin	9016
High St	Mosgiel	9023
George	Dunedin	9016

$F = \{ST \to P, P \to T\}$

$\mathbf{F^+} = \{ \text{.........} \}$

Post code	Town
9016	Dunedin
9023	Mosgiel

$F = \{P \to T$ and trivial dependencies$\}$

Post code	Street
9023	High St
9016	High St
9016	George

$F = \{$trivial dependencies only$\}$

Testing preservation of dependencies

Consider R = (A B C D)

with F = { A → B, B → C, C → D, D → A}

with a decomposition R_1 (A B), R_2 (B C), R_3 (C D) and corresponding sets of dependencies F_1, F_2, F_3

Does this preserve the dependency D → A ?

Compute $\mathbf{F^+}$ and project it onto R_1 .. R_3, giving F_1 .. F_3

Then test if $\mathbf{F^+}$ is equivalent to $F_1 \cup F_2 \cup F_3$

Example—post codes

Street	Town	Post code
High St	Dunedin	9016
High St	Mosgiel	9023
George	Dunedin	9016

$F = \{ST \rightarrow P, P \rightarrow T\}$

$\mathbf{F^+} = \{ \ldots\ldots\ldots \}$

ST → P is lost!

Post code	Town
9016	Dunedin
9023	Mosgiel

$F = \{P \rightarrow T$ and trivial dependencies$\}$

Post code	Street
9023	High St
9016	High St
9016	George

$F = \{$trivial dependencies only$\}$

Relational data model components

- The three core components in the relational model:

- Objects or **relations**—the structure of data organisation
- **Integrity constraints**—enforcing constraints and rules
- **Operators**—manipulation of data

- Finally, let's look at **operators**

Codd’s Relational Algebra

- The original relational algebra as described by Codd defines eight operators, in two groups:

Set operators

- union
- intersection
- difference ('minus')
- Cartesian product (‘times')

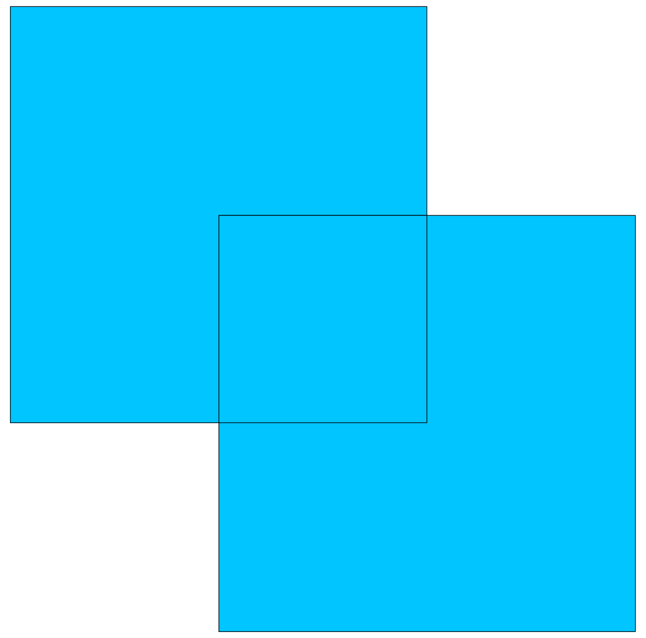
Special relational operators

- restrict (or ‘select’)
- project
- join
- divide

Overview—set operators

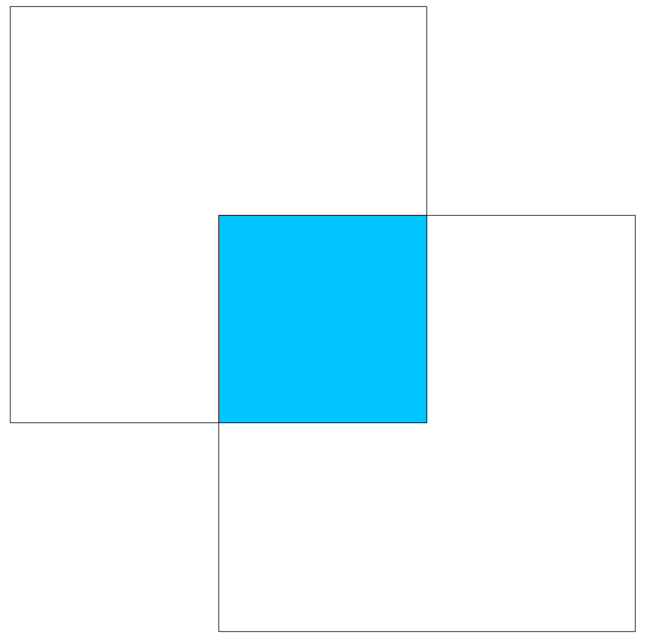
Union

$R \cup S$



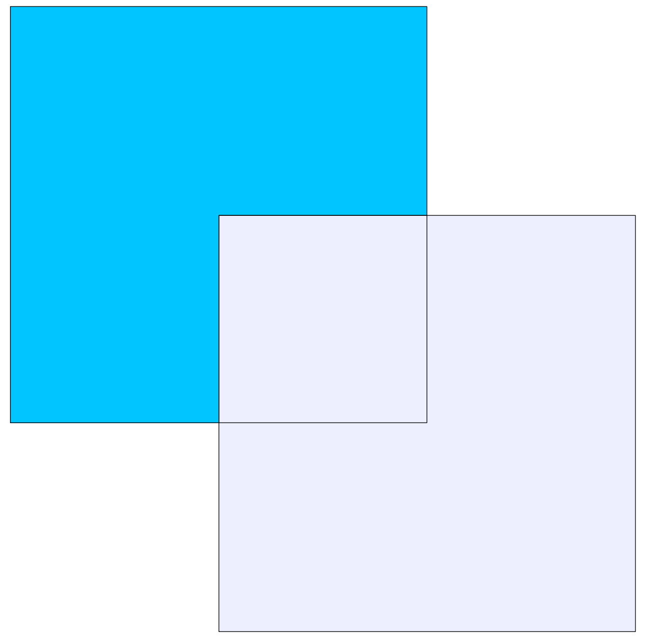
Intersection

$R \cap S$



Difference

$R - S$



Overview—set operators

- Cartesian product

R × ***S*** → ***Q***

A	B
a1	b1
a2	b2
a3	b3

C	D
c1	d1
c2	d2

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b2	c1	d1
a2	b2	c2	d2
a3	b3	c1	d1
a3	b3	c2	d2

Overview—join

Join $R \bowtie_{B=BB} S \rightarrow Q$

R:

A	B
a1	b1
a2	b1
a3	b2
a4	b3

S:

BB	C
b1	c1
b2	c2

Q:

A	B	BB	C
a1	b1	b1	c1
a2	b1	b1	c1
a3	b2	b2	c2

Natural join $R \bowtie S \rightarrow Q$

R:

A	B
a1	b1
a2	b1
a3	b2
a4	b3

S:

B	C
b1	c1
b2	c2

Q:

A	B	C
a1	b1	c1
a2	b1	c1
a3	b2	c2

Overview — division

- Division can be expressed as a sequence of operations using just project, Cartesian product, and difference

R ÷ ***S*** → ***Q***

R

A	B
a1	b1
a2	b1
a1	b2
a4	b2

S

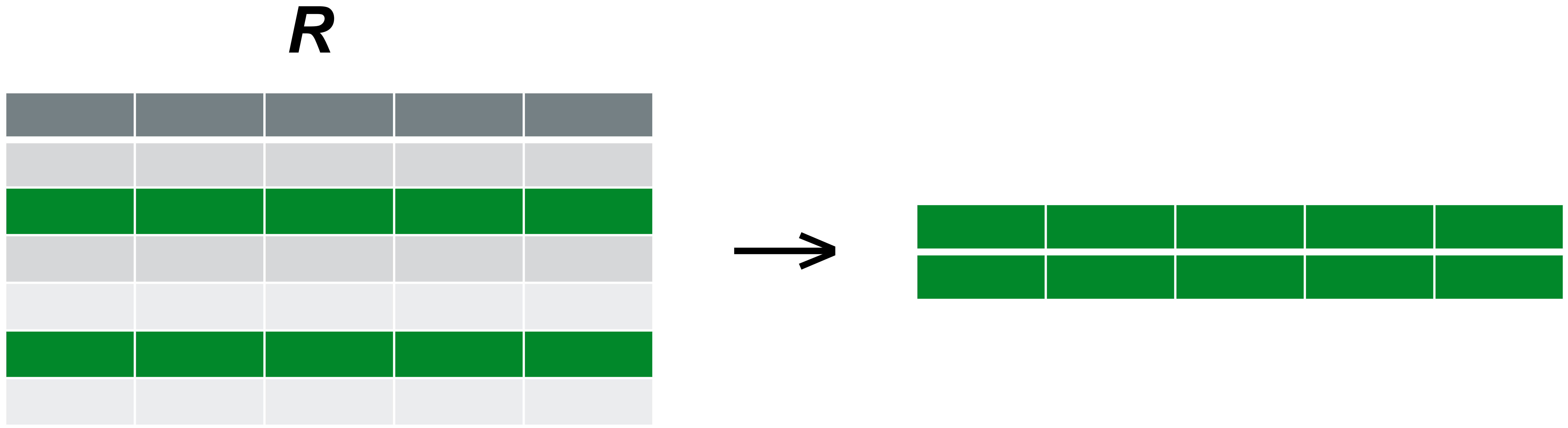
B
b1
b2

Q

A
a1

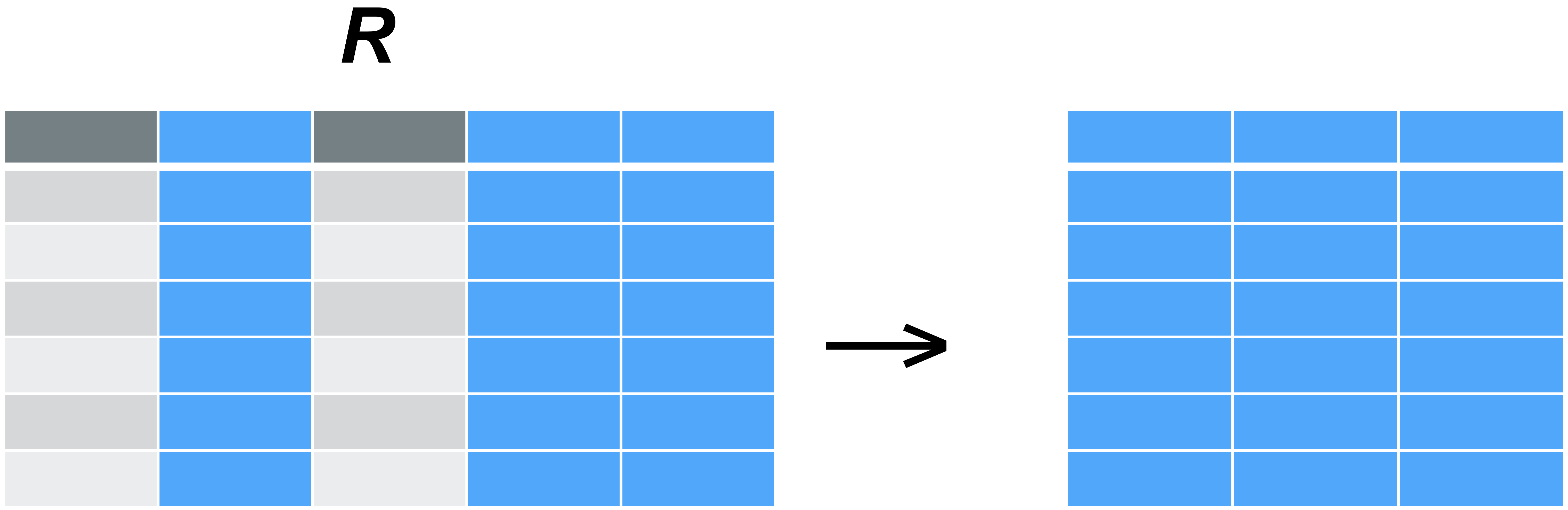
Special relational operators—restriction

- Notation: $\sigma_{condition}(R)$
  - Works on a single relation *R*, selecting the subset of the tuples of *R* that satisfy the given condition
- Produces a horizontal partition of *R*



Special relational operators—projection

- Notation: $\Pi_{attribute\_list}(R)$
  - Works on a single relation *R*, defining a new relation containing the specified attribute list from *R* (eliminating duplicate tuples)
- Produces a vertical partition of *R*



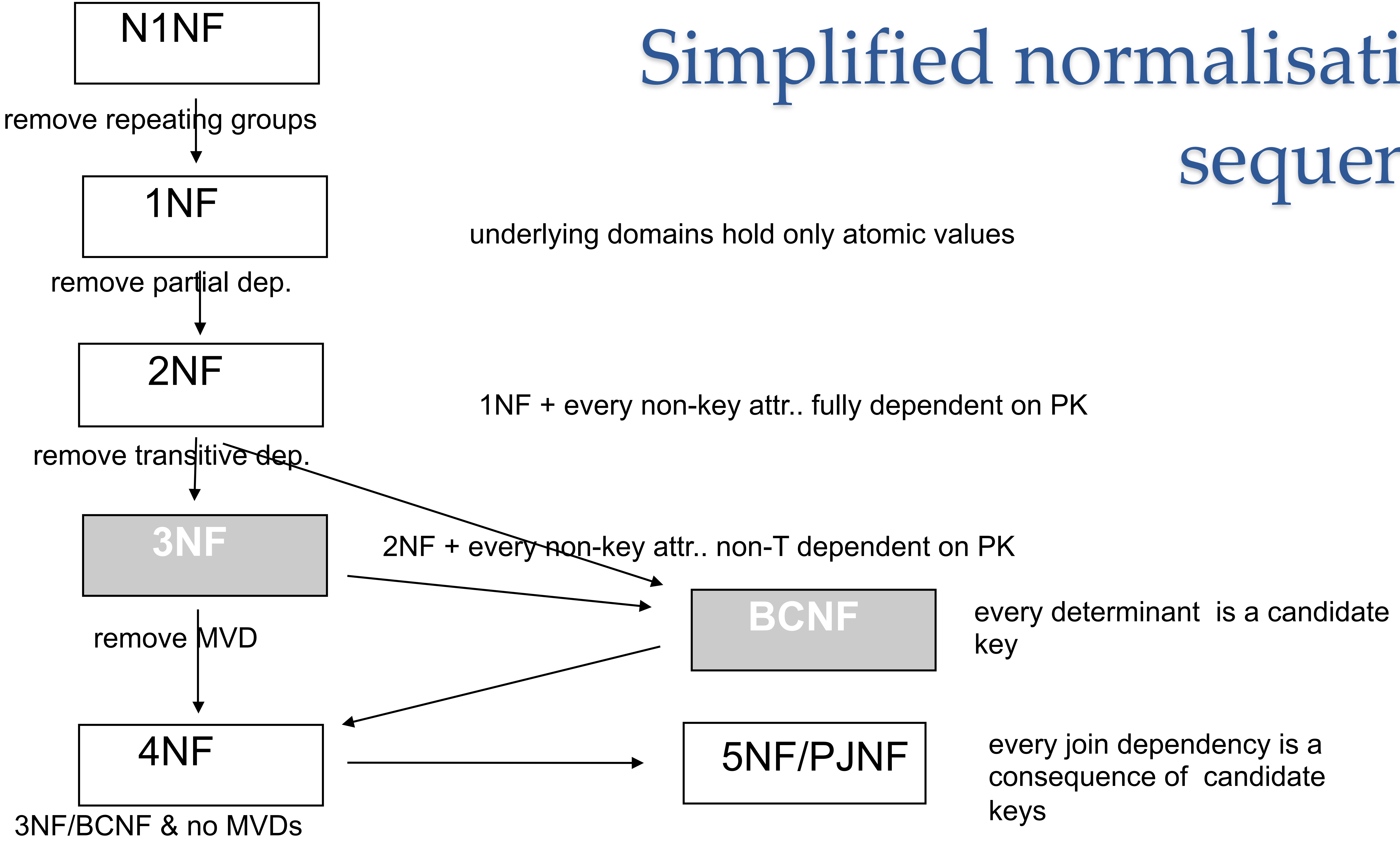
Formal notation

- UNION $\cup$
- DIFFERENCE $-$
- TIMES (Cartesian product) $\times$
- RESTRICT (Select) σ
- PROJECT π
- INTERSECT $\cap$
- JOIN $\bowtie$
- DIVIDE $\div$
- Note: The first five operators are primitive; others can be represented as sequence of primitive operators

Normalisation

- (And now for something completely different!)
- Normalisation = simplification
  - "A step by step reversible process of replacing a given collection of relations by successive collections in which the relations have a progressively simpler and more regular structure." —E.F. Codd

Simplified normalisation sequence



Normal forms—from 1NF to BCNF

- **First Normal Form (1NF)**
  - A relation is in 1NF if the domains of all attributes contain only atomic values and the value of any attribute in a tuple is a single value from the domain
- **Second Normal Form (2NF)**
  - A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the whole candidate key (assumes only one CK)
- **Third Normal Form (3NF)**
  - A relation is in 3NF if it is in 2NF and every non-prime attribute is non-transitively dependent on every key
- **Boyce-Codd Normal Form (BCNF)**
  - Every determinant (i.e., left hand side of a functional dependency) is a candidate key

Definition of **prime attribute**: an attribute that occurs in some candidate key

General definitions of 3NF & BCNF

Consider a relation *R* and a set of FDs

R is in **3NF** if, for every non-trivial FD X→A in *R*,

either (a) X is a superkey of *R*

or (b) A is a prime attribute of *R*

R is in **BCNF** if X is a superkey of *R*

Note that most relations in 3NF are also in BCNF

Example—post codes

Street	**Town**	**Post code**
High St	Dunedin	9016
High St	Mosgiel	9023
George	Dunedin	9016

$$\begin{array}{cc} X \rightarrow A & X \rightarrow A \\ F = \{ST \rightarrow P, & P \rightarrow T\} \end{array}$$

candidate key = ST

R is 3NF but not BCNF

3NF if:
X is a superkey of *R*
or A is a prime attribute of *R*

Normalisation by Decomposition or Synthesis

- For any relation there is always a dependency-preserving, non-loss decomposition/synthesis into a set of relations in 3NF

- There is always a non-loss decomposition/synthesis into BCNF, not always dependency-preserving.

Decomposition into 3NF

- Consider a relational scheme CTHRSG

  where C = course (paper)
  T = teacher
  H = hour (time of day)
  R = room
  S = student number
  G = grade

  and F = {C→T, HR→C, HT→R, CS→G, HS→R}

- Suggest a candidate key and the normal form of the relation

Candidate keys? Normal form?

- R(CTHRSG) *F* = {C→T, HR→C, HT→R, CS→G, HS→R}
- Some random keys to test (better to use an algorithm!)
  - CST? No—can't generate attribute H (also not minimal)
  - HRS? No—not minimal so not candidate key, because HS→R
  - HS? **Yes**—attribute closure is CTHRSG, and can't remove H or S
- Which normal form: 1NF, 2NF, 3NF ? At least 1NF, but…
  - **2NF**: 1NF, and all non-prime attributes (CTRG) depend on whole of all candidate keys (there is only one candidate key: HS)
  - Not 3NF: e.g., T is only transitively dependent on HS (via C)

Decomposition into 3NF

- Given: a relation scheme *R*, with a (minimal) set of dependencies, *F*:
- Any attributes of *R* not involved in *F*? Eliminate from *R* to form a separate relation scheme
- If one of the dependencies in *F* involves all the attributes of *R*, then *R* itself is in 3NF
- Otherwise the decomposition consists of a scheme XA for each dependency $X \rightarrow A$ in *F*
- For set of dependencies $X \rightarrow A_1$, $X \rightarrow A_2, \ldots X \rightarrow A_n$, combine to form the scheme $X \rightarrow A_1 A_2 \ldots A_n$

Decomposition into 3NF

R (CTHRSG)

F = {C→T, HR→C, HT→R, CS→G, HS→R}

F is a minimal cover and the algorithm leads to:

R_1 (CT)

R_2 (HRC)

R_3 (HTR)

R_4 (CSG)

R_5 (HSR)

Some additional 3NF decomposition steps

For a set of dependencies $X \rightarrow A_1$, $X \rightarrow A_2, \ldots X \rightarrow A_n$, combine to form the scheme $X \rightarrow A_1 A_2 \ldots A_n$

If our minimal cover was

R (ABCD) $F = \{A \rightarrow B, A \rightarrow C, D \rightarrow B\}$

Then the algorithm leads directly to

R_1 (ABC) $F = \{A \rightarrow BC\}$

R_2 (DB) $F = \{D \rightarrow B\}$

Decomposition into BCNF (simplified)

If *F* holds the FDs of the relation *R*

and $X \rightarrow A$ holds in *R*

and X is not a superkey of *R*

and A is not in X

then decompose *R* into:

R_1 (XA) R_1 is in BCNF

R_2 (R - A) R_2 becomes *R*; continue decomposition

ref: Ullman: Principles of Database and Knowledge-base systems, vol 1, CSP, 1988

Decomposition into BCNF

R (CTHRSG)

F = { C→T, HR→C, HT→R, CS→G, HS→R }

R_1 (CT) key=C, F_1 = { C→T }

R_Z (CHRSG) key=HS,

F_Z = { HR→C, CS→G, HS→R, *HC→R* }

R_2 (CHR) keys=HR,CH F_2 = { HR→C, CH→R }

R_Z (CHSG) key=HS, F_Z = { CS→G, *HS→C* }

and… R_3 (CSG) key=CS

and… R_4 (HSC) key=HS

Projected dependencies

Compare 3NF & BCNF decomposition

- R (CTHRSG) $F = \{ C \to T, HR \to C, HT \to R, CS \to G, HS \to R \}$

3NF	R_1 (CT)
	R_2 (HRC)
	R_3 (HTR)
	R_4 (CSG)
	R_5 (HSR)

BCNF	R_1 (CT)	key=C,	$F_1 = \{C \to T\}$
	R_2 (CHR)	keys=HR,CH	$F_2 = \{HR \to C, CH \to R\}$
	R_3 (CSG)	key=CS	$F_3 = \{CS \to G\}$
	R_4 (HSC)	key=HS	$F_4 = (HS \to C\}$

HT → R is lost in BCNF!

Overall objectives of normalisation

- Eliminate certain kinds of redundancy
- Avoid certain update anomalies
- Produce a design that is:
  - a ‘good’ representation of the real world
  - intuitively easy to understand and is a good base for future growth
- Simplify enforcement of certain integrity constraints

Summary

- The relational model of data
- Functional dependencies give semantic meaning to relations
- Knowing some FDs allows us to infer other FDs
- The rules for this are called Armstrong's axioms
- FDs can be treated in a rigorous formal manner, just like the underlying relational theory

Summary—2

- Normalisation:
  - is a step by step reversible process
  - is used to simplify data
  - provides multiple levels of simplification
  - allows us to remove redundancy in data while maintaining information present in original table
- Most relation schemes are in 3NF or BCNF
- 3NF ignores dependencies between candidate keys
- BCNF does not necessarily preserve dependencies