Let's start

SCC.201
Database Management Systems

2023 - Week 4 – Relational Algebra

Uraz C Turker & Ricki Boswell

What will you learn today?

- Relational algebra
- Some SQL code.

Curriculum Design: Outline Syllabus

This module builds upon knowledge gained in Part I by providing a theoretical background to the design, implementation and use of database management systems, both for data designers and application developers. It takes into account all relevant aspects related to information security in the design, development and use of database systems. The course consists of a number of related sections, which range from single lectures to multi-lecture streams, depending on the required depth of coverage. The sections are as follows.

Introduction : we begin with a brief history of how the need for database management systems (DBMS) grew over time and how they are applied in day to day scenarios.

Database Design: before making use of a DBMS, we must capture our requirements : what data do we actually wish to model? We make use of the Extended Entity-Relationship (EER) model which is both a technique and a notation for designing the data in a DBMS independent way.

The Relational Model: now the de-facto standard for DBMS, this was a revolutionary step taken in 1970. We extensively examine the Model, looking at relational database systems, the model itself and the normalisation process, the relational algebra (the mathematical theory that underpins the model), the three schema architecture and schema definition in SQL. Finally, we look at how we can map the EER model into an equivalent Relational Model. The resultant database is then examined in terms of access rights and privileges.

A (re)Introduction to SQL: SQL is the de-facto standard for DBMS query languages. We look at both the DDL (data definition language) and DML (data manipulation language). We introduce the use of views, a powerful mechanism for providing privacy and security. We look at the Discretionary Access Control (DAC) features that allow the granting and withholding of access rights and privileges.

Accessing relational DBMS via Java: we explore the facilities of the JDBC and show how we can write applications in Java which connect with a relational DBMS (in practice, MySQL).

The Physical Model: as Computer Scientists, our students need an awareness of the techniques that allow rapid access to stored data. In this section, we examine the physical data organisation and associated access methods. We show under what circumstances the organisations can be applied, and we look at how queries can be optimised.

Transaction processing and concurrency control: a huge part of DBMS in practice is the need to support transactions and concurrency, allowing huge numbers of users to access the DBMS at any one time while still ensuring the consistency of the data. This stream examines the problems and solutions in depth.

Lecture 1	Introduction to the module, Why do we need Databases? Entity Relationship Model
Lecture 2	Entity Relationship Model (ERM) (cont.)
Lab	**A gentle start to the ER diagrams.**
Lecture 1	Relational Model (RM)
Lecture 2	ER to RM
Lab	**ER diagrams.**
Lecture 1	Relational Model To SQL & SQL scripting
Lecture 2	**Review**
Lab	**ER to Relational Model.**
Lecture 1	**Relational Algebra**
Lecture 2	**Functional Dependencies + 1st, 2nd Normal Forms**
Lab	**Relational Algebra + SQL + Relational Model To SQL.**
Lecture 1	**3dr and Boycott normal forms. Advanced SQL queries.**
Lecture 2	JDBC
Lab	**Functional dependencies and Normalisation.**
Lecture 1	Physical Storage - record files
Lecture 2	Storage - secondary files
Lab	**Functional dependencies and Normalisation + JDBC Example.**
Lecture 1	Record Search - B-Trees
Lecture 2	Search - Hashing
Lab	**Working on project**
Lecture 1	Access Routines
Lecture 2	Query Optimisation
Lab	Project Grade Phase 1
Lecture 1	Concurrency - Transaction Processing
Lecture 2	Locking
Lab	Project Grade Phase 2
Lecture 1	Advanced SQL - schemas, views, access control
Lecture 2	Review and recap?
Lab	Project Grade Phase 3

Query Languages

- For manipulation and retrieval of stored data
- Relational model supports simple yet powerful query languages
- Query languages are not as complex as programming languages
- They are specialized for data manipulation and retrieval

Relational Algebra

- It is a mathematical query language
- Forms the basis of the SQL query language
- Relational Calculus is another mathematical query language but it is declarative rather than operational
- We will concentrate on relational algebra in this course

Basics of Querying

- A query is applied to relation instances, and the result of a query is also a relation instance.

eid	ename	Salary	age
28	Eric	90K	35
58	Kyle	100K	33

→

ename	Salary
Eric	90K
Kyle	100K

Relational Algebra Operations

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
- *Cross-product* ($\times$) Combines two relations.
- *Set-difference* ($-$) Tuples in relation 1, but not in relation 2.
- *Union* ($\cup$) Tuples in relation 1 and in relation 2.

Relational Algebra

- Additional operations:
  - Intersection,
  - *Join*
  - division,
  - Renaming
- Each operation returns a relation therefore operations can be *composed*

Projection

- Input is a single relation instance
- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the input relation.
- Projection operator has to eliminate *duplicates* (Why??)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(S2)$$

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

Projection

- Input is a single relation instance
- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the input relation.
- Projection operator has to eliminate *duplicates* (Why??)
  - Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it. (Why not?)

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{age}(S2)$$

age
35.0
55.5

Selection

- Input is a single relation instance
- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of input relation.

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\sigma_{rating > 8}(S2)$$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

Selection

- Input is a single relation instance
- Selects rows that satisfy *selection condition*.
- No duplicates in result! (Why?)
- *Schema* of result identical to schema of input relation.
- *Result* relation can be the *input* for another relational algebra operation!

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

sname	rating
yuppy	9
rusty	10

Union, Intersection, Set-Difference

- All of these operations take two input relations, which must be *union-compatible*:
  - Same number of fields.
  - `Corresponding’ fields have the same type.

Union

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$S1 \cup S2$

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

Intersection

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$S1 \cap S2$

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

Set Difference

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$S1 - S2$

sid	sname	rating	age
22	dustin	7	45.0

Cross-Product

- S1 X R1
- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 **X** *R1*

(sid)	sname	rating	age	(sid)	bid	day

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 **X** *R1*

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 **X** *R1*

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1* X *R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 **X** *R1*

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96

Cross-Product

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Both S1 and R1 have a field called *sid*. Which may cause a conflict when referring to columns

S1* X *R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Renaming Operator

Takes a relation schema and gives a new name to the schema and the columns

$$\rho\ (C(1 \rightarrow sid1, 5 \rightarrow sid2),\ S1 \times R1)$$

C

1	2	3	4	5	6	7
sid1	sname	rating	age	sid2	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Joins

- *Condition Join* : $R \bowtie_{c} S = \sigma_{c}(R \times S)$

  - *Result schema* same as that of cross-product.
  - Fewer tuples than cross-product, might be able to compute more efficiently
  - Sometimes called a *theta-join*.

Joins

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$\sigma_{S1.sid < R1.sid}(S1 \times R1)$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$\sigma_{S1.sid < R1.sid}(S1 \times R1)$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities.***

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{sid} R1$$

sid	sname	rating	age	bid	day

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities*.**

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{sid} R1$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities*.**

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{sid} R1$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96

[redacted]

Equi-Join: A special case of condition join where the condition *c* contains only ***equalities***.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie_{sid} R1$$

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

Joins

- *Equi-Join*: A special case of condition join where the condition *c* contains only ***equalities.***

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

- *Result schema* similar to cross-product, but only one copy of fields for which equality is specified.

$$S1 \bowtie_{sid} R1$$

- *Natural Join*: Equijoin on ***all* common** fields.

$$S1 \bowtie R1$$

Division

- Not supported as a primitive operator, but useful for expressing queries like:
  *Find Players who have played **all** games*.
- Let A have 2 fields, x and y; B have only field y:
  - A/B = Keeps x values providing following condition

Division

- Not supported as a primitive operator, but useful for expressing queries like:
  *Find Players who have played **all** games*.
- Let *A* have 2 fields, *x* and *y*; *B* have only field *y*:
  - *A/B* = Keeps *x* values providing following condition
    - For *B* there exist *x* in *A* such that *B* **X** *x* is a member of *A*

Division

- Not supported as a primitive operator, but useful for expressing queries like:
  *Find Players who have played **<u>all</u>** games*.
- Let *A* have 2 fields, *x* and *y*; *B* have only field *y*:
  - *A/B* = Keeps *x* values providing following condition
    - For *B* there exist *x* in *A* such that *B* **X** *x* is a member of *A*
  - i.e., ***A/B* contains all *x* values (players) such that for *<u>every</u>* *y* value (game) in *B*, there is an *x-y* paired value in *A*.**
  - *Or*: If the set of *y* values (games) associated with an *x* value (player) in *A* contains all *y* values in *B*, the *x* value is in *A/B*.
- In general, *x* and *y* can be any lists of fields; *y* is the list of fields in *B*, and *x* U *y* is the list of fields of *A*.

Examples of Division A/B

- For *B (one attribute)* there exist *x* in *A* such that *B* **X** *x* is a member of *A*

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

Find names of sailors who've reserved boat #103

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

Find names of sailors who've reserved boat #103

$$\pi_{sname}(\sigma_{bid=103}(\mathrm{Re}serves \bowtie Sailors))$$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

Find names of sailors who've reserved boat #103

$$\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

Find names of sailors who've reserved boat #103

$$\pi_{sname}(\sigma_{bid=103}(\underbrace{Reserves \bowtie Sailors}))$$

sid	*sname*	*rating*	*age*	*bid*	*day*
22	Dustin	7	45.0	101	10/10/98
22	Dustin	7	45.0	102	10/10/98
22	Dustin	7	45.0	103	10/8/98
22	Dustin	7	45.0	104	10/7/98
31	Lubber	8	55.5	102	11/10/98
31	Lubber	8	55.5	103	11/6/98
31	Lubber	8	55.5	104	11/12/98
64	Horatio	7	35.0	101	9/5/98
64	Horatio	7	35.0	102	9/8/98
74	Horatio	9	35.0	103	9/8/98

Result of Natural Join

Find names of sailors who've reserved boat #103

$$\pi_{sname}(\underbrace{\sigma_{bid=103}(\text{Reserves} \bowtie \text{Sailors})}_{\text{Result of Selection bid=103}})$$

Result of Selection bid=103

sid	sname	rating	age	bid	day
22	Dustin	7	45.0	101	10/10/98
22	Dustin	7	45.0	102	10/10/98
22	Dustin	7	45.0	103	10/8/98
22	Dustin	7	45.0	104	10/7/98
31	Lubber	8	55.5	102	11/10/98
31	Lubber	8	55.5	103	11/6/98
31	Lubber	8	55.5	104	11/12/98
64	Horatio	7	35.0	101	9/5/98
64	Horatio	7	35.0	102	9/8/98
74	Horatio	9	35.0	103	9/8/98

Find names of sailors who've reserved boat #103

$$\underbrace{\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))}$$

Result of Projection on sname

sid	*sname*	*rating*	*age*	*bid*	*day*
22	Dustin	7	45.0	101	10/10/98
22	Dustin	7	45.0	102	10/10/98
22	$\boxed{\text{Dustin}}$	7	45.0	103	10/8/98
22	Dustin	7	45.0	104	10/7/98
31	Lubber	8	55.5	102	11/10/98
31	$\boxed{\text{Lubber}}$	8	55.5	103	11/6/98
31	Lubber	8	55.5	104	11/12/98
64	Horatio	7	35.0	101	9/5/98
64	Horatio	7	35.0	102	9/8/98
74	$\boxed{\text{Horatio}}$	9	35.0	103	9/8/98

Find names of sailors who've reserved boat #103

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R2* of Reserves

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie \ Sailors)$

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R2* of Reserves

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103}\mathrm{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid
22
31
74

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$

- Solution 2: $\rho\ (Temp1,\ \sigma_{bid=103}\ \text{Reserves})$

  $\rho\ (Temp2,\ Temp1 \bowtie Sailors)$

  $\pi_{sname}\ (Temp2)$

- Solution 3: $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

Find names of sailors who've reserved a red boat

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance $B1$ of Boats

Find names of sailors who've reserved a red boat

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance $B1$ of Boats

Find names of sailors who’ve reserved a red boat

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance $B1$ of Boats

Find names of sailors who've reserved a red boat

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B*1 of Boats

Find names of sailors who’ve reserved a red boat

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance $B1$ of Boats

Find names of sailors who've reserved a red boat

22
31
64

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B*1 of Boats

Find names of sailors who've reserved a red boat

22
31
64

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B*1 of Boats

Find names of sailors who've reserved a red boat

- NOTE: Information about boat colour is only available in Boats; so need an extra join:

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \mathrm{Re}serves \bowtie Sailors)$$

- A more efficient solution:

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie \mathrm{Re}s) \bowtie Sailors)$$

Find names of sailors who've reserved a red <u>and</u> a green boat.

HOW ABOUT THIS ANSWER?

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B*1 of Boats

$$\rho(Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \mathrm{Reserves}))$$

$$\rho\ (Tempgreen, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie \mathrm{Reserves}))$$

$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

Find names of sailors who've reserved a red <u>and</u> a green boat.

HOW ABOUT THIS ONE?

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance $S3$ of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance $R2$ of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance $B1$ of Boats

$$\rho(Tmp1, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \text{Reserves}))$$

$$\rho(Tmp2, \pi_{sid}((\sigma_{color='green'} Boats) \bowtie \text{Reserves}))$$

$$\pi_{sname}(Tmp1 \bowtie Sailors) \cap \pi_{sname}(Tmp2 \bowtie Sailors)$$

Find the names of sailors who've reserved all boats

Division

$$\pi_{sid,bid} Reserves$$

sid	bid
22	101
22	102
22	103
22	104
31	102
31	104
64	101
64	102
74	103

$$(\pi_{sid,bid} Reserves)/(\pi_{bid} Boats)$$

$$\pi_{sname}(Division \bowtie Sailors)$$

sid
22

$$\pi_{bid} Boats$$

bid
101
102
103
104

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S3* of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R2* of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 4.17 An Instance *B1* of Boats

Find the names of sailors who've reserved all boats

- Uses division; schemas of the input relations must be carefully chosen:

$$\rho\ (Tempsids, (\pi_{sid,bid}\text{Re}serves) \,/\, (\pi_{bid}Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

- To find sailors who've reserved all 'Interlake' boats:

$$\ldots\ldots \quad / \ \pi_{bid}(\sigma_{bname='Interlake'}Boats)$$

Let's start

SCC.201
Database Management Systems

2023 - Week 4 – Relational Algebra – Schema Refinement

Uraz C Turker & Ricki Boswell

Relational Algebra Operations

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
-
-
-
-

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Relational Algebra Operations

Lancaster University

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
-
-
-

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Relational Algebra Operations

sid	bid	day
22	101	10/10/98
22	102	10/10/98

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
- *Cross-product* ($\times$) Combines two relations.
-
-

sid	bid	day	bid	bname	color
22	101	10/10/98	101	Interlake	blue
22	102	10/10/98	102	Interlake	red
22	101	10/10/98	103	Clipper	green
22	102	10/10/98	104	Marine	red
22	101	10/10/98	102	Interlake	red
22	102	10/10/98	101	Interlake	blue
22	101	10/10/98	104	Marine	red
22	102	10/10/98	103	Clipper	green

Relational Algebra Operations

bid
101
102

bid
101
102
103
104

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
- *Cross-product* ($\times$) Combines two relations.
- *Set-difference* ($-$) Tuples in relation 1, but not in relation 2.
-

Relational Algebra Operations

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Basic operations:

- *Selection* (σ) Selects a subset of rows from relation.
- *Projection* (π) Deletes unwanted columns from relation.
- *Cross-product* ($\times$) Combines two relations.
- *Set-difference* ($-$) Tuples in relation 1, but not in relation 2.
- *Union* ($\cup$) Tuples in relation 1 and in relation 2.
- Join: Theta-Join, Equi-Join, Natural-Join.

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

Relational Algebra Operations

Basic operations:

-*Selection* (σ) Selects a subset of rows from relation.

-*Projection* (π) Deletes unwanted columns from relation.

-*Cross-product* ($\times$) Combines two relations.

-*Set-difference* ($-$) Tuples in relation 1, but not in relation 2.

-*Union* ($\cup$) Tuples in relation 1 and in relation 2.

-Join: Theta-Join (conditional), Equi-Join (Selected fields have same value), Natural-Join (All common fields have same value).

Relational Algebra Operations

Basic operations:

-*Selection* (σ) Selects a subset of rows from relation.

-*Projection* (π) Deletes unwanted columns from relation.

-*Cross-product* ($\times$) Combines two relations.

-*Set-difference* ($-$) Tuples in relation 1, but not in relation 2.

-*Union* ($\bigcup$) Tuples in relation 1 and in relation 2.

-Join: Theta-Join (conditional), Equi-Join (Selected fields have same value), Natural-Join (All common fields have same value).

-/ Division!

(Practical)
Relational Algebra-SQL relation.

- A standard for querying relational data
- Basic query structure

SELECT	[DISTINCT]	*attribute-list*
FROM	*relation-list*	
WHERE	*condition*	

- DISTINCT is an optional keyword indicating that duplicates should be eliminated. (Otherwise duplicate elimination is not done)

```
sqlite> Select DISTINCT D.NoOfEmp FROM Department D WHERE Dname = "45";
244
sqlite>
```

SQL

- A standard for querying relational data
- Basic query structure

```
SELECT    [DISTINCT]  attribute-list
FROM      relation-list
WHERE     condition
```

- Conditions (ATTR *op* CONST or ATTR1 *op* ATTR2, where *op* is one of ($<, >, =, \leq, \geq, \neq$) combined using AND, OR and NOT.

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:

```
SELECT      [DISTINCT]  attribute-list
FROM        relation-list
WHERE       condition
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.

```
SELECT    [DISTINCT] attribute-list
FROM      relation-list
WHERE     condition
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.
  - Discard resulting tuples if they do not satisfy the *conditions*.

```
SELECT    [DISTINCT]  attribute-list
FROM      relation-list
WHERE     condition
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.
  - Discard resulting tuples if they do not satisfy the *conditions*.
  - Display attributes that are in *attribute-list*.

```
SELECT      [DISTINCT]  attribute-list
FROM        relation-list
WHERE       condition
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.
  - Discard resulting tuples if they do not satisfy the *conditions*.
  - Display attributes that are in *attribute-list*.
  - If DISTINCT is specified, eliminate duplicate rows.
-

```
SELECT   [DISTINCT]  attribute-list
FROM     relation-list
WHERE    condition
```

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.
  - Discard resulting tuples if they do not satisfy the *conditions*.
  - Display attributes that are in *attribute-list*.
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimiser will find more efficient strategies to compute *the same answers*.

```
SELECT    [DISTINCT]  attribute-list
FROM      relation-list
WHERE     condition
```

Example of Conceptual Evaluation

SELECT sname
FROM Sailors, Reserves
WHERE Sailors.sid=Reserves.sid AND Reserves.bid=103

Query: Find names of sailors who Reserved boat number 103

$$\pi_{sname}((\sigma_{bid=103} Reserves) \bowtie Sailors)$$

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 4.15 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 4.16 An Instance *R*2 of Reserves

Range Variables

Query: Find names of sailors who Reserved boat number 103

```
SELECT  sname
FROM    Sailors, Reserves
WHERE   Sailors.sid=Reserves.sid AND Reserves.bid=103
```

```
SELECT  S.sname
FROM    Sailors S, Reserves R
WHERE   S.sid=R.sid AND bid=103
```

Range variables are necessary when joining a table with itself !!!

Expressions and Strings

```
SELECT  S.age, age1=S.age-5, 2*S.age AS age2
FROM  Sailors S
WHERE  S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two new fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*

Expressions and Strings

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two new fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*
- **AS and = are two ways to name fields in result.**
-

Expressions and Strings

```
SELECT  S.age, age1=S.age-5, 2*S.age AS age2
FROM  Sailors S
WHERE  S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two new fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*
- **AS and = are two ways to name fields in result.**
- **LIKE is used for string matching. `_' stands for any one character and `%' stands for 0 or more arbitrary characters.**

Find sid's of sailors who've reserved a red or a green boat

B

<u>BID</u>	Colr
b1	red
b2	grn

R

SID	<u>BID</u>
s1	b1
s1	b2
s2	b1

- UNION: Can be used to compute the union of any two *union-compatible* sets of tuples (which are themselves the result of SQL queries).

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
  AND (B.color='red' OR B.color='green')
```

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
         AND B.color='red'
UNION
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
         AND B.color='green'
```

- EXCEPT : Used to compute the set difference of two *union-compatible* sets of tuples
- What do we get if we replace **UNION** with **EXCEPT** in the previous SQL query?

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
           AND B.color='red'
EXCEPT
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid
           AND B.color='green'
```

A

-

B

B

BID	Colr
b1	red
b2	grn

R

SID	BID
s1	b1
s1	b2
s2	b1

Example

EMPLOYEE(NAME, <u>SSN</u>, BDATE, ADDRESS, SALARY)

DEPARTMENT(DNAME, <u>DNUMBER</u>, MGRSSN)

(MGRSSN references SSN in EMPLOYEE table)

WORKSIN(<u>ESSN, DNUMBER</u>, HOURS)

(DNUMBER references DNUMBER in DEPARTMENT table)

(ESSN reference SSN in EMPLOYEE table)

Write the relational algebra expressions for the following queries:

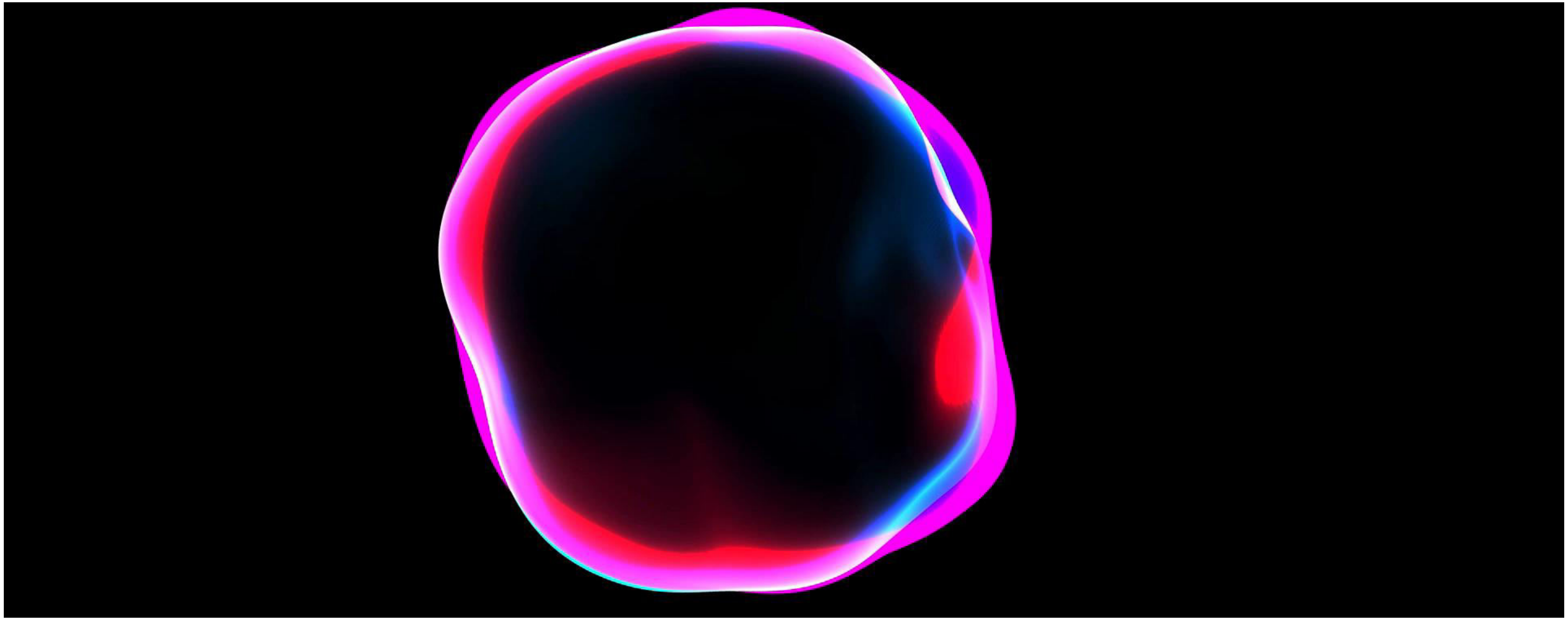
1) List the names of employees whose salary is greater than 30000

2) List the names of employees who work in “shoes” department

3) List the names of employees who work in all departments

More will be provided after Normalisation

Normal forms



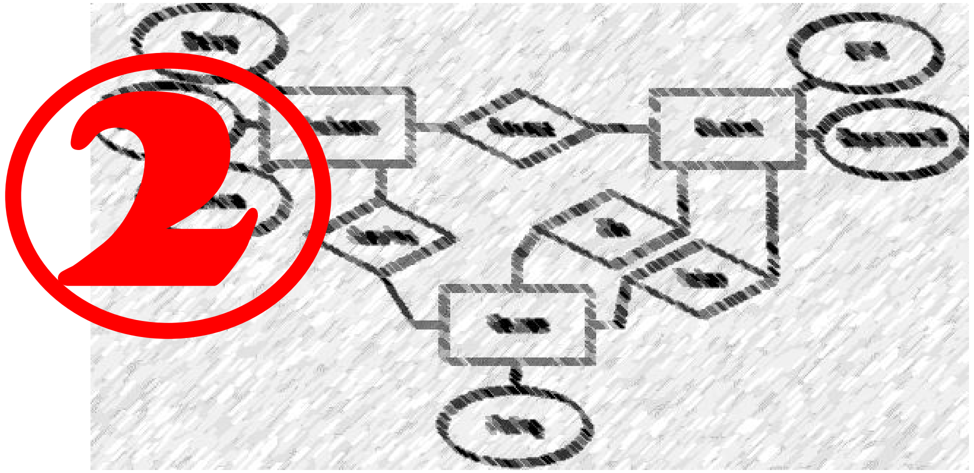
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.

1

- We received a work plan in plain English.
-
-
-
-

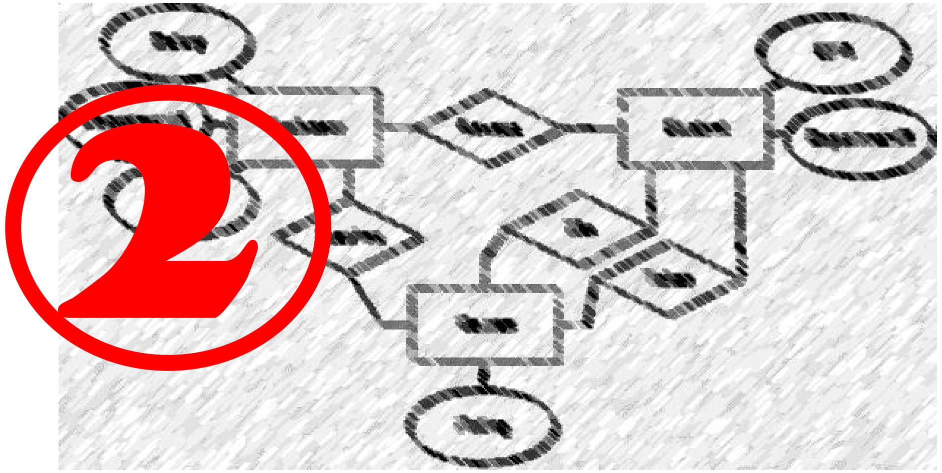
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.

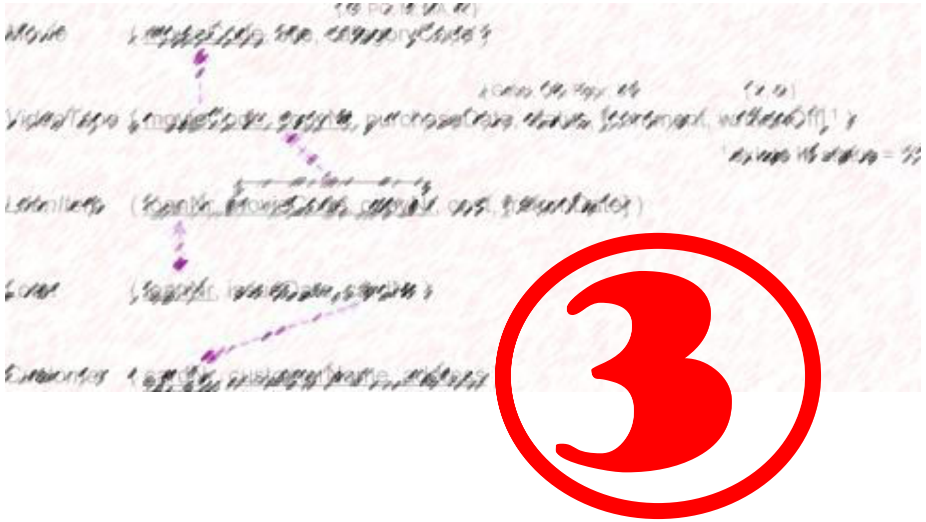
1



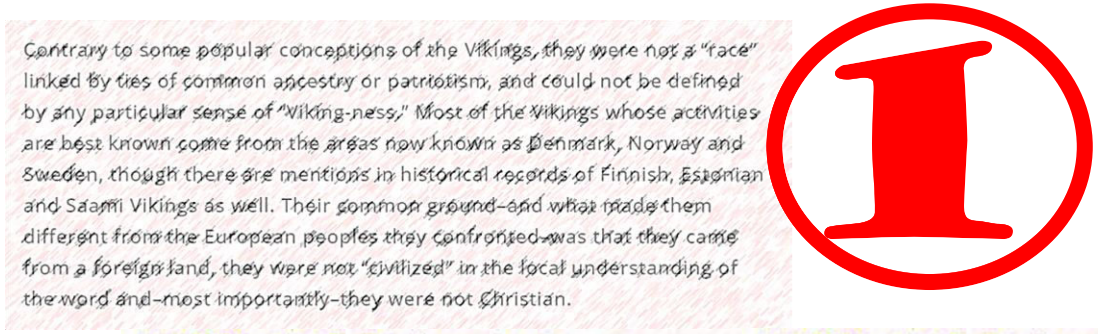
- We received a work plan in plain English.
- We derived its ER diagram.
-
-
-

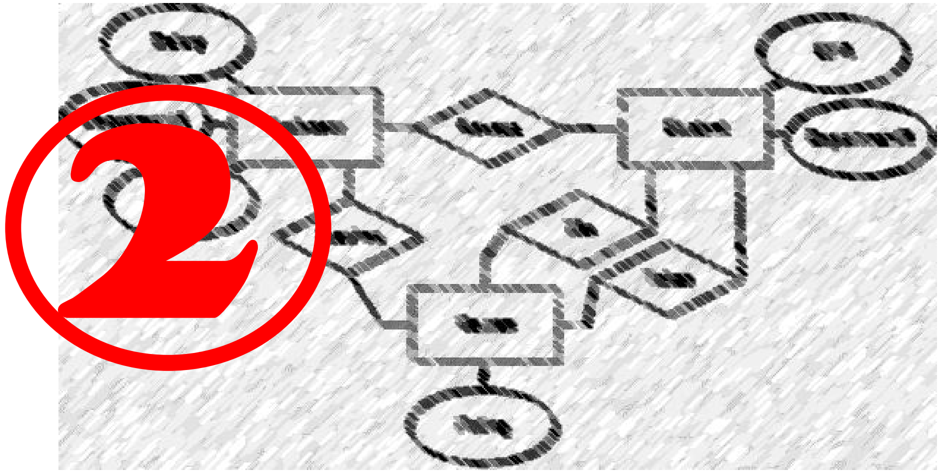
Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.

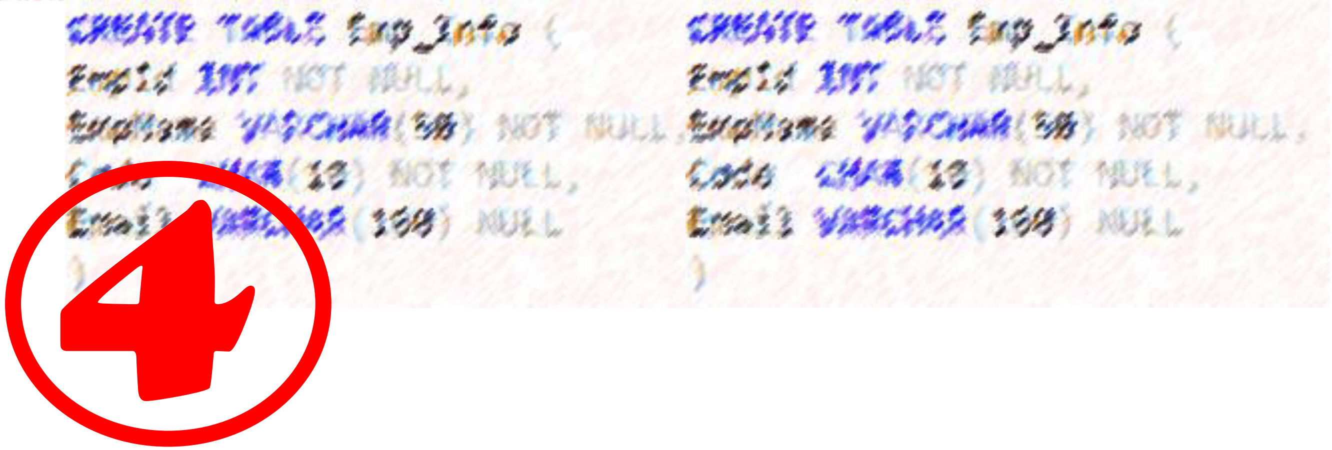


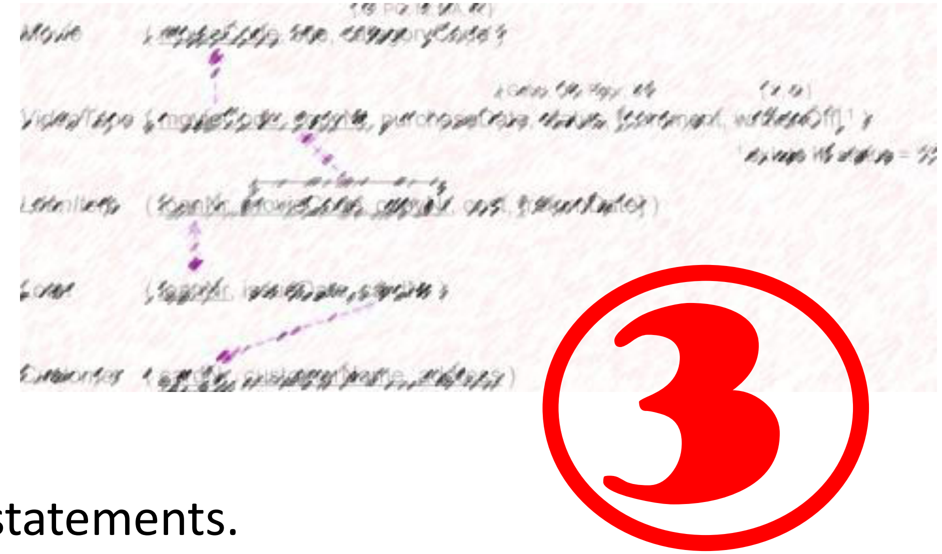


- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
-
-

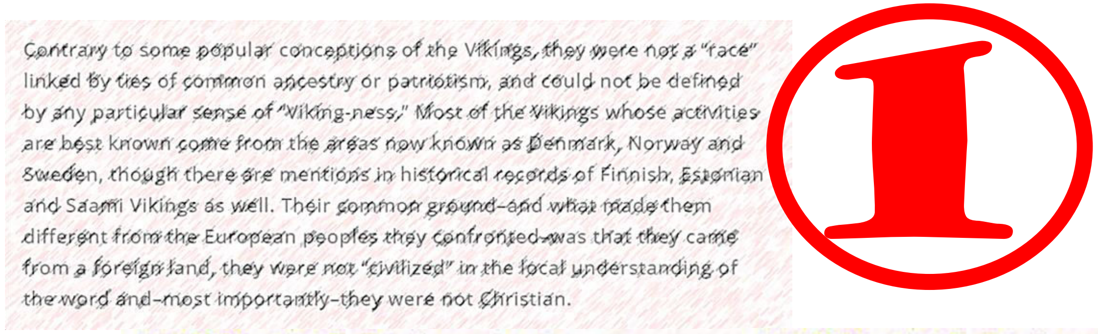


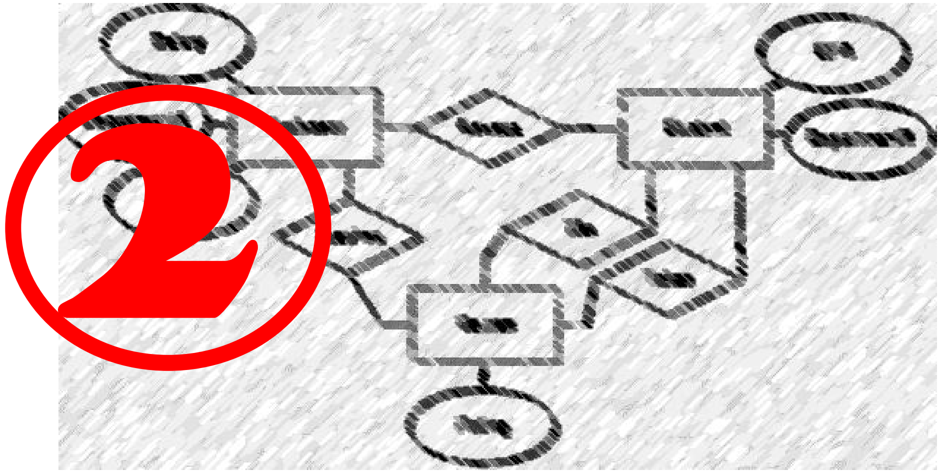


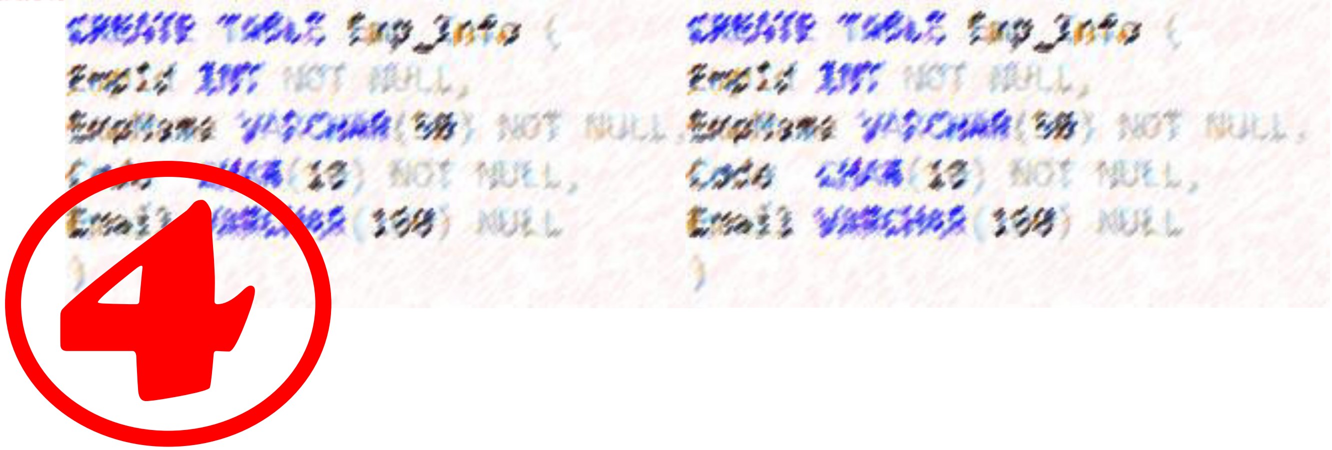


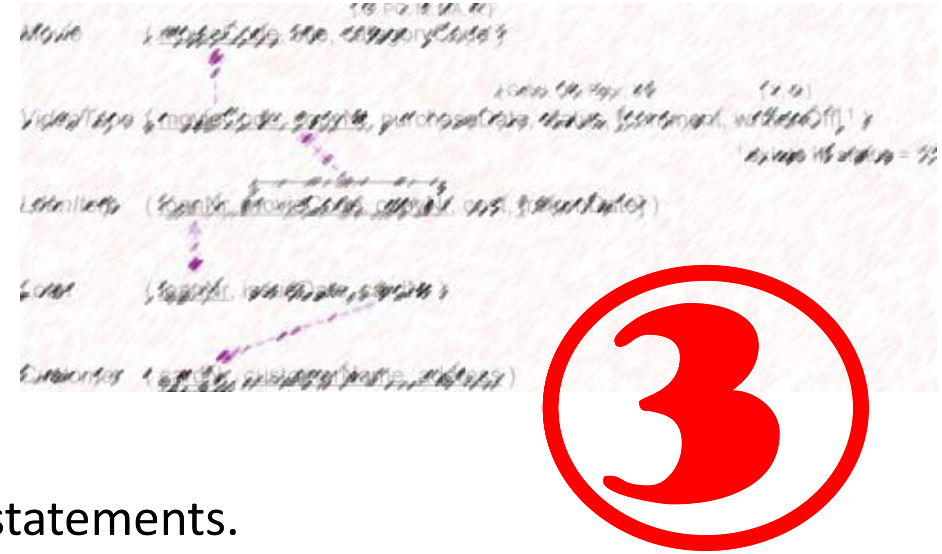


- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
- We created the tables in a database using DDL statements.
-


Contrary to some popular conceptions of the Vikings, they were not a "race" linked by ties of common ancestry or patriotism, and could not be defined by any particular sense of "Viking-ness." Most of the Vikings whose activities are best known come from the areas now known as Denmark, Norway and Sweden, though there are mentions in historical records of Finnish, Estonian and Saami Vikings as well. Their common ground–and what made them different from the European peoples they confronted–was that they came from a foreign land, they were not "civilized" in the local understanding of the word and–most importantly–they were not Christian.







- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
- We created the tables in a database using DDL statements.
- We insert some tuples.

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40

- We received a work plan in plain English.
- We derived its ER diagram.
- We created its Relational Schema and ICs.
- We created the tables in a database using DDL statements.
- We insert some tuples.

ssn	name	lot	rating	hourly_wages	hours_worked
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40

<u>*ssn*</u>	*name*	<u>*lot*</u>	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40
341-12-1124	Thomas	54	8	9	23

- Anomalies
  - Insertion anomalies
    - Recording wrong *hourly_wages*

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
612-67-4134	Uraz	39	8	10	40

What is the hourly wage when the rating is 5?

- Anomalies
  - Insertion anomalies
    - Recording wrong *hourly_wages*
  - Deletion anomalies
    - If we delete rating, we also lose the *hourly_wages* info.
  -

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Hasan	48	8	12	40
231-31-5368	Robert	22	8	12	30
131-24-3650	Ercan	36	5	12	30
434-26-3751	Fox	38	5	12	32
612-67-4134	Uraz	39	8	12	40

- Anomalies
  - Insertion anomalies
    - Recording wrong *hourly_wages*
  - Deletion anomalies
    - If we delete rating, we also lose the *hourly_wages* info.
  - Modification (update) anomalies

ssn	name	lot	rating	hourly_wages	hours_worked
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40

ssn	name	lot	rating	hourly_wages	hours_worked
123-22-3666	Hasan	48	8	10	40
231-31-5368	Robert	22	8	10	30
131-24-3650	Ercan	36	5	7	30
434-26-3751	Fox	38	5	7	32
612-67-4134	Uraz	39	8	10	40

ssn	name	lot	rating	hours_worked
123-22-3666	Hasan	48	8	40
231-31-5368	Robert	22	8	30
131-24-3650	Ercan	36	5	30
434-26-3751	Fox	38	5	32
612-67-4134	Uraz	39	8	40

rating	hourly_wages
8	10
5	7

DECOMPOSITION

Decomposition

Decompositions are done to remove redundant data that can lead to anomalies.

- There are levels of redundancy which are determined by Normal Forms.

Normal Forms

- Normal forms are standards for a good DB schema (introduced by Codd in 1972)
- If a relation is in a particular normal form (such as BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimised.
- Normal forms help us decide if decomposing a relation helps.

Normal Forms

- First Normal Form: No set valued attributes (only atomic values)

sid	name	phones
1	ali	{5332344568, 2165533561}
2	veli	…
3	ayse	…
4	fatma	…

Normal Forms

- 2nd Normal form
- Prime attribute: any attribute that is part of a key WITHIN CANDIDATE KEY
- Non-prime attributes: rest of the attributes

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Normal Forms

- 2^{nd} Normal form
- Prime attribute: any attribute that is part of a key WITHIN CANDIDATE KEY
- Non-prime attributes: rest of the attributes

Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Normal Forms

- 2^{nd} Normal form
- Prime attribute: any attribute that is part of a key WITHIN CANDIDATE KEY
- Non-prime attributes: rest of the attributes

Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.

Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Normal Forms

- 2nd Normal form
- Prime attribute: any attribute that is part of a key WITHIN CANDIDATE KEY
- Non-prime attributes: rest of the attributes

Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.

Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.

And the MANAGER is a NONPRIME ATTRIBUTE.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Normal Forms

Second Normal Form: Every non-prime attribute should be ***fully functionally dependent*** on the whole part of every key (i.e., candidate keys).

What does Functional Dependent mean?

Functional Dependencies (FORMAL)

- A <u>functional dependency</u> between attributes X, Y (X → Y) holds over relation R if, for every allowable instance *r* of R:
  - $t1 \in r,\ t2 \in r,$ *such that* $\pi_X(t1) = \pi_X(t2)$ implies $\pi_Y(t1) = \pi_Y(t2)$
  - i.e., given two tuples in *r*, if the X values agree, then the Y values must also agree.

	X	Y	Z
t1	1	a	p
	2	b	q
t2	1	a	r
	2	b	p

Whenever X value of a tuple is *1*, Y value of the same tuple is *a*
Whenever X value of a tuple is *2*, Y value of the same tuple is *b*

Functional Dependencies

Does the following relation instance satisfy FD X->Y ?

X	Y	Z
1	a	p
2	b	q
1	a	r
3	b	p

Functional Dependencies

If X is a member of a candidate key, we have FD X -> Y Z (trivial dependency)

X	Y	Z
1	a	p
2	b	q
1	a	p
3	b	p

Example

- Some FDs on Hourly_Emps:
  - *ssn* is the key: S $\longrightarrow$SNLRWH
  - *rating* determines *hrly_wages*: R$\longrightarrow$W

Did you notice anything wrong with the following instance ?

S	N	L	R	W	H
			1	100	
			2	200	
			3	250	
			2	300	

Example

- Some FDs on Hourly_Emps:
  - *ssn* is the key: S $\longrightarrow$ SNLRWH
  - *rating* determines *hrly_wages*: R $\longrightarrow$ W

Did you notice anything wrong with the following instance ?

S	N	L	R	W	H
			1	100	
			2	200	
			3	250	
			2	300	

Example

- Some FDs on Hourly_Emps:
  - *ssn* is the key: S $\longrightarrow$ SNLRWH
  - *rating* determines *hrly_wages*: R $\longrightarrow$ W

Did you notice anything wrong with the following instance ?

S	N	L	R	W	H
			1	100	
			2	200	
			3	250	
			2	200	

Normal Forms

Second Normal Form: Every attribute not part of a key (non-prime attribute) should be ***fully functionally dependent*** on the whole part of every key (i.e., candidate keys).

A relation is in 2NF if it is in 1NF, and **every non-prime attribute of the relation depends on the whole of every candidate key**. Note that it does not restrict the non-prime to non-prime attribute dependency. (Part of a key should not decide non-prime attribute)

Normal Forms

Second Normal Form: Every attribute not part of a key (non-prime attribute) should be ***fully functionally dependent*** on the whole part of every key (i.e., candidate keys).

A relation is in 2NF if it is in 1NF, and **every non-prime attribute of the relation depends on the whole of every candidate key**. Note that it does not restrict the non-prime to non-prime attribute dependency. (Part of a key should not decide non-prime attribute)

To check:

Normal Forms

Second Normal Form: Every attribute not part of a key (non-prime attribute) should be ***fully functionally dependent*** on the whole part of every key (i.e., candidate keys).

A relation is in 2NF if it is in 1NF, and **every non-prime attribute of the relation depends on the whole of every candidate key**. Note that it does not restrict the non-prime to non-prime attribute dependency. (Part of a key should not decide non-prime attribute)

To check:

1 Find the candidate key,

Normal Forms

Second Normal Form: Every attribute not part of a key (non-prime attribute) should be ***fully functionally dependent*** on the whole part of every key (i.e., candidate keys).

A relation is in 2NF if it is in 1NF, and **every non-prime attribute of the relation depends on the whole of every candidate key**. Note that it does not restrict the non-prime to non-prime attribute dependency.

To check:

1 Find the candidate key,

2 Check if a non-prime attribute functionally depends on some parts of a candidate key.

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.
- Observe:

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.
- Observe:
  - If PROJECT_ID = {45} or {23}, MANAGER is Mr.X,

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.
- Observe:
  - If PROJECT_ID = {45} or {23}, MANAGER is Mr.X,
  - Else If PROJECT_ID=67, MANAGER is Mr.Z,

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.
- Observe:
  - If PROJECT_ID = {45} or {23}, MANAGER is Mr.X,
  - Else If PROJECT_ID=67, MANAGER is Mr.Z,
  - Else If PROJECT_ID=78, MANAGER is Mr.Y.
-

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

2nd Normal form

- Let us assume that the Candidate Key of the relation is {(EMP_ID,PROJECT_ID)}.
- Therefore EMP_ID and PROJECT_ID are PRIME ATTRIBUTEs.
- And the MANAGER is a NONPRIME ATTRIBUTE.
- Observe:
  - If PROJECT_ID = {45} or {23}, MANAGER is Mr.X,
  - Else If PROJECT_ID=67, MANAGER is Mr.Z,
  - Else If PROJECT_ID=78, MANAGER is Mr.Y.
- So Project_Id->Manager. There is PARTIAL DEPENDENCY, so the relation is not in the 2nd Normal Form.

EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

Normalise

EMP_ID	PROJECT_ID	PROJECT
1	23	Extract
1	67	Load
2	45	Extract
3	78	Transform
3	23	Extract
4	23	Extract
5	78	Transform
5	67	Load

PROJECT_ID	MANAGER
23	Mr.X
45	Mr.X
78	Mr.Y
67	Mr.Z

2^{nd} Normal Form

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1^{st} Normal Form?

2nd Normal Form

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes

2nd Normal Form

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find Candidate KEY

2nd Normal Form

<u>Supplier</u>	<u>Part#</u>	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find Candidate KEY
  - { (S,P) }

2nd Normal Form

Lancaster University

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find Candidate KEY
  - { (S,P) }
- Check if a non-prime attribute is functionally dependent on some parts of a candidate key.

2nd Normal Form

Supplier	Part#	Location	Stock
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find Candidate KEY
  - { (S,P) }
- Check if a non-prime attribute is functionally dependent on some parts of a candidate key.
  - S -> L
-

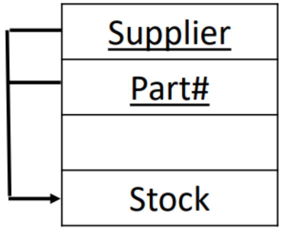
2nd Normal Form

Supplier	*Part#*	*Location*	*Stock*
Acme	1	London	17
Acme	2	London	25
Acme	3	London	17
Ajax	1	Bristol	25
Ajax	3	Bristol	18
Amco	1	Glasgow	3
Amco	2	Glasgow	22
Jamco	1	Glasgow	3

- Is this in 1st Normal Form?
  - Yes
- Find Candidate KEY
  - { (S,P) }
- Check if a non-prime attribute is functionally dependent on some parts of a candidate key.
  - S -> L
- Not in 2nd normal form.

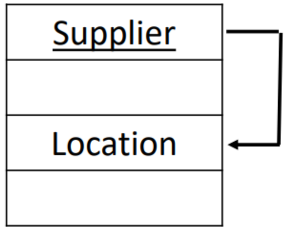
Normalise

- As a part (S) of a candidate key (S,P) implies NON-PRIME ATTRIBUTE {L}, we need to normalise the relation by introducing relations (S,P,St) and (S,L).



Supplier	Part#	Stock
Acme	1	17
Acme	2	25
Acme	3	17
Ajax	1	25
Ajax	3	18
Amco	1	3
Amco	2	22
Jamco	1	3

We give each functional dependency its own relation.



Supplier	Location
Acme	London
Ajax	Bristol
Amco	Glasgow
Jamco	Glasgow

Third Normal Form (3NF)

- Relation R is in 3NF if it is in 2NF and for all $X \rightarrow A$
  - $A \in X$ (called a *trivial* FD), or
  - X contains a key for R, or
  - A is part of some key for R.
  - In other words, **there should not be the case that another non-prime attribute determines a non-prime attribute.**
- If R is in 3NF, some redundancy still is possible. i.e. a part of a key determines some other attribute.

3NF

- E# is the key and Candidate key is {(E#)}.

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2^{nd} NF).
- How about FDs of non-prime attributes?

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
- How about FDs of non-prime attributes?
  - M#->MTEL# (or MTEL#->M#)
-

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- E# is the key and Candidate key is {(E#)}.
- Every attribute is atomic, and there does not exist a non-prime attribute that is partially implied by part of a key in the candidate key (So it is in 2nd NF).
- How about FDs of non-prime attributes?
  - M#->MTEL# (or MTEL#->M#)
- So it is not in 3NF.

E#	Name	Age	Job#	M#	MTEL#
100	J. Smith	22	22	1	64413
101	J. Smith	45	22	1	64413
102	A. Adams	22	17	2	37611
103	K. Bedford	35	12	3	18653
104	F. Bloggs	55	17	3	18653
108	A. Adams	35	12	3	18653
109	J. Dean	35	12	1	64413

3NF

- To normalise, for every FD we create a new table.
  - M#->MTEL# (or MTEL#->M#)
  - E#-> N,A,J#,M#

E#	Name	Age	Job#	M#
100	J. Smith	22	22	1
101	J. Smith	45	22	1
102	A. Adams	22	17	2
103	K. Bedford	35	12	3
104	F. Bloggs	55	17	3
108	A. Adams	35	12	3
109	J. Dean	35	12	1

M#	MTEL#
1	64413
1	64413
2	37611
3	18653
3	18653
3	18653
1	64413