Informatics 1: Data & Analysis

Lecture 5: Relational Algebra

Ian Stark

School of Informatics
The University of Edinburgh

Tuesday 26 January 2016
Semester 2 Week 3

Support (1/3) !

If you have questions about something in the lectures, difficulties with tutorial exercises, or want to find out more on the material, ask someone.

- Other students: in your tutorial group, in the lab, elsewhere.
- InfBASE: Drop-in helpdesk, staffed by tutors, open each day in FH-1B.19, see web for timetable.
- Your course tutor: in person at your tutorials, or by email.
- The lecturer, Ian Stark: in person after lectures, drop-in office hour IF 5.04 every Wednesday 1030–1130, or by email.
- Online: Piazza Q+A; Facebook page and groups

Support (2/3) !

Interaction

Engage with the course and other students

Course Web	Lecture log; slides; tutorial exercises. **http://blog.inf.ed.ac.uk/da15**
Piazza	Help others and ask your own questions. Follow links from course web pages to sign up.
Facebook Page	Follow **http://fb.me/inf1da16**
Facebook Group	Join **https://www.facebook.com/groups/965727066816763**

Support (3/3) !

For technical support when machines aren't working or you have problems with software on DICE, fill out the computing support form.

http://computing.help.inf.ed.ac.uk

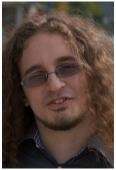
For administrative support in anything related to teaching, contact the Informatics Teaching Organisation (ITO) by filling out their online contact form, or go to the ITO office in Forrest Hill.

http://www.inf.ed.ac.uk/teaching/contact

If you are having difficulties affecting all of your courses, or issues arising outside the University, contact your personal tutor.

Tutors !

Georgoulas



Ikonomov

Reijsbergen

Peternek

Dragan

Marina

Martinov

Almeida

Quesada Real

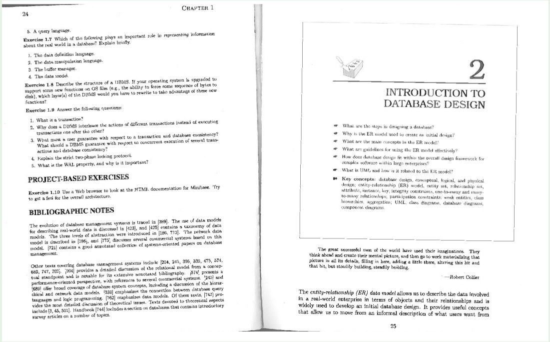
Zalewski

Thorne

Stark

TopHat: Assigned Reading

R. Ramakrishnan and J. Gehrke.
Database Management Systems.
McGraw-Hill, third edition, 2003.



24 CHAPTER 1

5. A query language.

Exercise 1.7 Which of the following plays an important role in *representing* information about the real world in a database? Explain briefly.

1. The data definition language.
2. The data manipulation language.
3. The buffer manager.
4. The data model.

Exercise 1.8 Describe the structure of a DBMS. If your operating system is upgraded to support some new functions on OS files (e.g., the ability to force some sequence of bytes to disk), which layer(s) of the DBMS would you have to rewrite to take advantage of these new functions?

Exercise 1.9 Answer the following questions:

1. What is a transaction?
2. Why does a DBMS interleave the actions of different transactions instead of executing transactions one after the other?
3. What must a user guarantee with respect to a transaction and database consistency? What should a DBMS guarantee with respect to concurrent execution of several transactions and database consistency?
4. Explain the strict two-phase locking protocol.
5. What is the WAL property, and why is it important?

PROJECT-BASED EXERCISES

Exercise 1.10 Use a Web browser to look at the HTML documentation for Minibase. Try to get a feel for the overall architecture.

BIBLIOGRAPHIC NOTES

The evolution of database management systems is traced in [289]. The use of data models for describing real-world data is discussed in [423], and [425] contains a taxonomy of data models. The three levels of abstraction were introduced in [186, 712]. The network data model is described in [186], and [775] discusses several commercial systems based on this model. [721] contains a good annotated collection of systems-oriented papers on database management.

Other texts covering database management systems include [204, 245, 305, 339, 475, 574, 689, 747, 762]. [204] provides a detailed discussion of the relational model from a conceptual standpoint and is notable for its extensive annotated bibliography. [574] presents a performance-oriented perspective, with references to several commercial systems. [245] and [689] offer broad coverage of database system concepts, including a discussion of the hierarchical and network data models. [339] emphasizes the connection between database query languages and logic programming. [762] emphasizes data models. Of these texts, [747] provides the most detailed discussion of theoretical issues. Texts devoted to theoretical aspects include [3, 45, 501]. Handbook [744] includes a section on databases that contains introductory survey articles on a number of topics.

2

INTRODUCTION TO DATABASE DESIGN

- What are the steps in designing a database?
- Why is the ER model used to create an initial design?
- What are the main concepts in the ER model?
- What are guidelines for using the ER model effectively?
- How does database design fit within the overall design framework for complex software within large enterprises?
- What is UML and how is it related to the ER model?
- **Key concepts:** database design, conceptual, logical, and physical design; entity-relationship (ER) model, entity set, relationship set, attribute, instance, key; integrity constraints, one-to-many and many-to-many relationships, participation constraints; weak entities, class hierarchies, aggregation; UML, class diagrams, database diagrams, component diagrams.

The great successful men of the world have used their imaginations. They think ahead and create their mental picture, and then go to work materializing that picture in all its details, filling in here, adding a little there, altering this bit and that bit, but steadily building, steadily building.

—Robert Collier

The *entity-relationship (ER) data model* allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design. It provides useful concepts that allow us to move from an informal description of what users want from

25

Q. Have you read this yet?

TopHat: ER Diagram

InfPALS peer-assisted learning is a student-to-student support service for first-year Informatics undergraduates.

Each InfPALS group is made of first-year students who meet up to work together, with a senior student facilitator. Students cannot be in more than one group, and some students aren't in InfPALS at all.

Diagram A:

Diagram B:

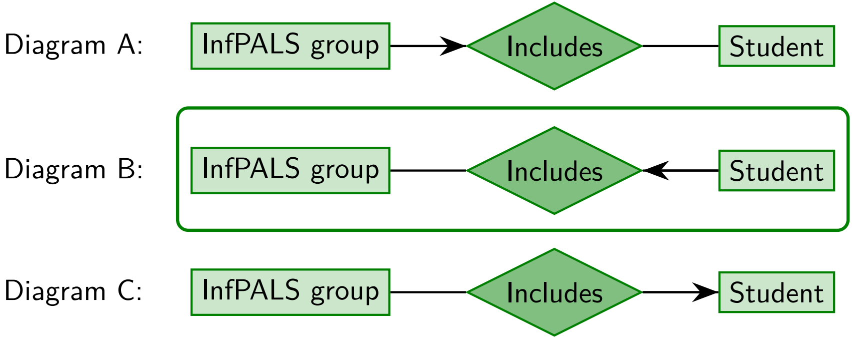
Diagram C:

Q. Which diagram captures the relationship between groups and students?

TopHat: ER Diagram

InfPALS peer-assisted learning is a student-to-student support service for first-year Informatics undergraduates.

Each InfPALS group is made of first-year students who meet up to work together, with a senior student facilitator. Students cannot be in more than one group, and some students aren't in InfPALS at all.



Q. Which diagram captures the relationship between groups and students?

Lecture Plan for Weeks 1–4

Data Representation

This first course section starts by presenting two common data representation models.

- The *entity-relationship (ER)* model
- The *relational* model

Note slightly different naming:
-relation**ship** vs. relation**al**

Data Manipulation

This is followed by some methods for manipulating data in the relational model and using it to extract information.

- *Relational algebra*
- The *tuple-relational calculus*
- The query language *SQL*

Remember Relations as Tables?

Relational databases take as fundamental the idea of a *relation*, comprising a *schema* and an *instance*.

Fields (a.k.a. attributes, columns)

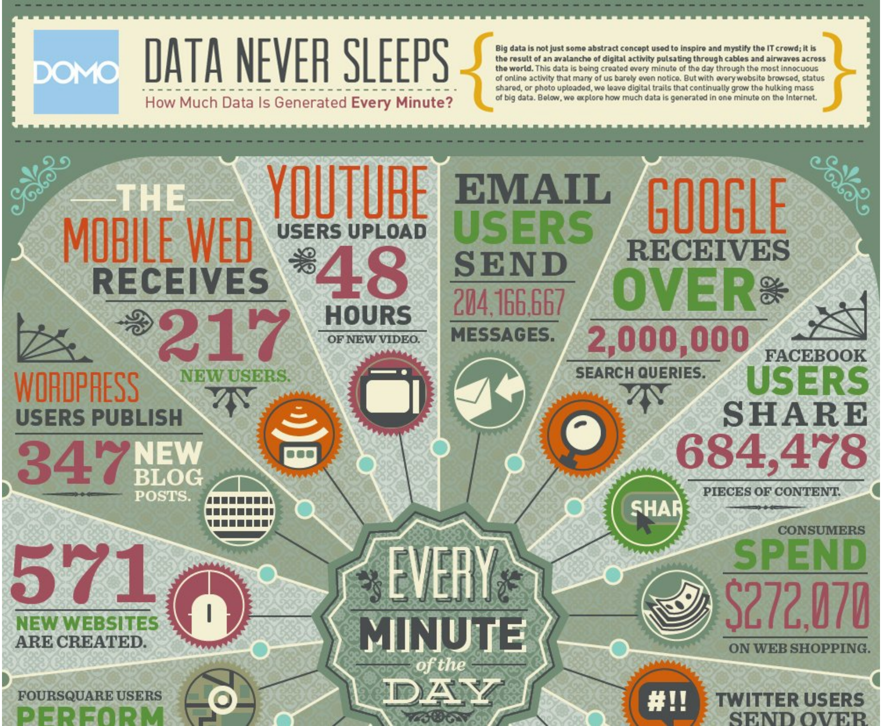
Schema

mn	name	age	email
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Tuples (a.k.a. records, rows)

Absolutely everything in a relational database is built from relations and operations upon them.

Every relational database is a linked collection of several tables like this: often much wider, and sometimes very, very much longer.


DOMO
DATA NEVER SLEEPS
How Much Data Is Generated Every Minute?
Big data is not just some abstract concept used to inspire and mystify the IT crowd; it is the result of an avalanche of digital activity pulsating through cables and airwaves across the world. This data is being created every minute of the day through the most innocuous of online activity that many of us barely even notice. But with every website browsed, status shared, or photo uploaded, we leave digital trails that continually grow the hulking mass of big data. Below, we explore how much data is generated in one minute on the Internet.
THE MOBILE WEB RECEIVES 217 NEW USERS.
YOUTUBE USERS UPLOAD 48 HOURS OF NEW VIDEO.
EMAIL USERS SEND 204,166,667 MESSAGES.
GOOGLE RECEIVES OVER 2,000,000 SEARCH QUERIES.
FACEBOOK USERS SHARE 684,478 PIECES OF CONTENT.
WORDPRESS USERS PUBLISH 347 NEW BLOG POSTS.
571 NEW WEBSITES ARE CREATED.
CONSUMERS SPEND $272,070 ON WEB SHOPPING.
EVERY MINUTE of the DAY
SHAR
#!!
FOURSQUARE USERS PERFORM
TWITTER USERS SEND OVER

The NIST Reference on Constants, Units, and Uncertainty

Table 5. SI prefixes

Factor	Name	Symbol	Factor	Name	Symbol
10^{24}	yotta	Y	10^{-1}	deci	d
10^{21}	zetta	Z	10^{-2}	centi	c
10^{18}	exa	E	10^{-3}	milli	m
10^{15}	peta	P	10^{-6}	micro	µ
10^{12}	tera	T	10^{-9}	nano	n
10^{9}	giga	G	10^{-12}	pico	p
10^{6}	mega	M	10^{-15}	femto	f
10^{3}	kilo	k	10^{-18}	atto	a
10^{2}	hecto	h	10^{-21}	zepto	z
10^{1}	deka	da	10^{-24}	yocto	y

Visual Networking Index IP Traffic Chart

IP Traffic Term	Equivalent	How much is that?
1 Petabyte	1,000 Terabytes or 250,000 DVDs	**100 Petabytes** The amount of data produced in a single minute by the new particle collider at CERN. **400 Terabytes** A digital library of all books ever written in any language
1 Exabyte	1,000 Petabytes or 250 million DVDs	**5 Exabytes** A transcript of all words ever spoken **100 Exabytes** A video recording of the all the meetings that took place last year across the world **150 Exabytes** The amount of data that has traversed the Internet since its creation **175 Exabytes** The amount of data that will cross the Internet in 2010 alone
1 Zettabyte	1,000 Exabytes or 250 billion DVDs	**66 Zettabytes** The amount of visual information conveyed from the eyes to the brain of the entire human race in a single year
1 Yottabyte	1,000 Zettabytes or 250 trillion DVDs	**20 Yottabytes** A holographic snapshot of the earth's surface

SHARE

RESEARCH ARTICLE

The World's Technological Capacity to Store, Communicate, and Compute Information

Martin Hilbert[1,*], **Priscila López**[2]

+ Author Affiliations

*To whom correspondence should be addressed. E-mail: mhilbert@usc.edu

Science 01 Apr 2011:
Vol. 332, Issue 6025, pp. 60-65
DOI: 10.1126/science.1200970

Article | Figures & Data | Info & Metrics | eLetters | PDF

You are currently viewing the abstract.

View Full Text

Abstract

We estimated the world's technological capacity to store, communicate, and compute information, tracking 60 analog and digital technologies during the period from 1986 to 2007. In 2007, humankind was able to store 2.9×10^{20} optimally compressed bytes, communicate almost 2×10^{21} bytes, and carry out 6.4×10^{18} instructions per second on general-purpose computers. General-purpose computing capacity grew at an annual rate of 58%. The world's capacity for bidirectional telecommunication grew at 28% per year, closely followed by the increase in globally stored information (23%). Humankind's capacity for unidirectional information diffusion through broadcasting channels has experienced comparatively modest annual growth (6%). Telecommunication has been dominated by digital technologies since 1990 (99.9% in digital format in 2007), and the majority of our technological memory has been in digital format since the early 2000s (94% digital in 2007).

View Full Text

Science

Vol 332, Issue 6025
01 April 2011

Table of Contents
Print Table of Contents
Advertising (PDF)
Classified (PDF)
Masthead (PDF)

ARTICLE TOOLS

- Email
- Download Powerpoint
- Print
- Save to my folders
- Alerts
- Request Permissions
- Citation tools
- Share

RELATED CONTENT

INTRODUCTION TO SPECIAL ISSUE
Challenges and Opportunities

SIMILAR ARTICLES IN:

- PubMed
- Google Scholar

CITED BY... +

CITING ARTICLES IN:

- Web of Science (160)
- Scopus (222)

RELATED JOBS FROM SCIENCECAREERS

- Computers, Mathematics

Information overload: There is so much data stored in the world that we may run out of ways to quantify it

- **Currently the largest measurement is a yottabyte**
- **New terminology can only be approved by the International Committee for Weights and Measures**

By CHARLES WALFORD FOR THE DAILY MAIL

PUBLISHED: 17:55, 12 December 2012 | **UPDATED:** 07:39, 13 December 2012

Not long ago, when talking about computer storage, the gigabyte was considered an enormous

Languages for Working with Relations

Once we have a quantity of structured data in the linked tables of a relational model we may want to rearrange it, build new data structures, and extract information through the use of *queries*.

To understand how this is done, we'll look at three interlinked languages:

Relational Algebra

> High-level mathematical operations for combining and processing relational tables.

Tuple-Relational Calculus

> A declarative mathematical notation for expressing queries over structured data.

SQL

> The standard programming language for writing queries on relational databases.

Relational Algebra

Relational algebra is a mathematical language for describing certain operations on the schemas and tables of a relational model. Each of these operations takes one or more tables, and returns another.

Basic operations:	selection σ, projection π, renaming ρ
	union $\cup$, difference $-$, cross-product $\times$
Derived operations:	intersection $\cap$ and different kinds of join $\bowtie$

Ted Codd gave a *completeness* proof showing that these operations were enough to express very general kinds of query: so, with an efficient implementation of these operations, you can answer all those queries.

Conversely, Codd's result also shows that to implement any expressive query language requires finding ways to carry out all of these operations.

Selection and Projection

mn	name	age	email
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Student

name	age
John	18
Mary	18
Helen	20
Peter	22

$\pi_{\text{name, age}}(\text{Student})$

mn	name	age	email
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

$\sigma_{\text{age}>18}(\text{Student})$

name	age
Helen	20
Peter	22

Combination

Selection picks out the rows of a table satisfying a logical predicate

Selection and Projection

mn	name	age	email
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Student

name	age
John	18
Mary	18
Helen	20
Peter	22

$\pi_{\text{name, age}}(\text{Student})$

mn	name	age	email
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

$\sigma_{\text{age}>18}(\text{Student})$

name	age
Helen	20
Peter	22

Combination

Projection picks out the columns of a table by their field name.

Selection and Projection

mn	name	age	email
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Student

name	age
John	18
Mary	18
Helen	20
Peter	22

$\pi_{\text{name, age}}(\text{Student})$

mn	name	age	email
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

$\sigma_{\text{age}>18}(\text{Student})$

name	age
Helen	20
Peter	22

Combination

Combining selection and projection picks out a rectangular subtable.

$$\pi_{\text{name,age}}(\sigma_{\text{age}>18}(\text{Student})) = \sigma_{\text{age}>18}(\pi_{\text{name,age}}(\text{Student}))$$

Definitions

Selection

Relation $\sigma_P(R)$ is the table of rows in R which satisfy *predicate* P.

Thus $\sigma_P(R)$ has the same schema as R, but possibly lower cardinality.

Predicates like P, Q, ... are made up of

- Assertions about field values: (age $>$ 18), (degree $=$ "CS"), ...
- Logical combinations of these: $(P \vee Q)$, $(P \wedge Q \wedge \neg Q')$, ...

Projection

Relation $\pi_{a_1,\ldots,a_n}(R)$ is the table of all tuples of the fields $a_1, \ldots, a_n$ taken from the rows of R.

Thus $\pi_{a_1,\ldots,a_n}(R)$ usually has a lower-arity schema than R, and may also have lower cardinality.

Logical Operators

Truth	TRUE	$\top$	T	tt	1			
Falsity	FALSE	$\bot$	F	ff	0			
Conjunction	AND	$P \wedge Q$	&	&&	$*$	$\cap$	$\bullet$	.
Disjunction	OR	$P \vee Q$	|	||	$+$	$\cup$		
Implication	IMPLIES	$P \Rightarrow Q$			$\rightarrow$	$\supset$		
Equivalence	IFF	$P \Leftrightarrow Q$			$\leftrightarrow$	$\equiv$		
Negation	NOT	$\neg P$	!	$\sim$				

Quantifiers $\forall, \exists$

Existential	EXISTS	$\exists$	?	$\exists x.P(x)$	$\exists x\ P(x)$	$\exists x(P(x))$
					$\exists x \in A \,.\, P(x)$	$\exists x : A \,.\, P(x)$
Universal	FORALL	$\forall$	!	$\forall x.P(x)$	$\forall x\ P(x)$	$\forall x(P(x))$
					$\forall x \in A \,.\, P(x)$	$\forall x : A \,.\, P(x)$

Set Comprehension

$$\{\, x \mid P(x) \,\} \qquad \{\, x \mid x \in A \wedge P(x) \,\} \qquad \{\, x \in A \mid P(x) \,\} \qquad \{\, x : A \mid P(x) \,\}$$

Compare Haskell list comprehension `[ x | x <- [1..20], even x ]`.

Brackets

$\Big(\quad \Big)$	Parentheses	Round brackets
$\Big[\quad \Big]$	Brackets	Square brackets
$\Big\{ \quad \Big\}$	Braces	Curly brackets
$\Big\langle \quad \Big\rangle$	Chevrons	Angle brackets

Colour Coding of Slides

Regular Substantive Slide

Selection

Relation $\sigma_P(R)$ is the table of rows in R which satisfy *predicate* P.

Thus $\sigma_P(R)$ has the same schema as R, but possibly lower cardinality.

Predicates like P, Q, ... are made up of

- Assertions about field values: $(\text{age} > 18)$, $(\text{degree} = \text{"CS"})$, ...
- Logical combinations of these: $(P \vee Q)$, $(P \wedge Q \wedge \neg Q')$, ...

Projection

Relation $\pi_{a_1,\dots,a_n}(R)$ is the table of all tuples of the fields $a_1, \dots, a_n$ taken from the rows of R.

Thus $\pi_{a_1,\dots,a_n}(R)$ usually has a lower-arity schema than R, and may also have lower cardinality.

/ January 26, 2016

Announcement Slide !

Reading

Before the next lecture, on Friday, read the remaining sections, §§2.5 onwards, of Ramakrishnan and Gehrke, completing Chapter 2.

These consider trade-offs and choices in the design of Entity-Relationship models, as well as more on the wider context of modelling.

Tutorial

Tutorial 1: Entity-Relationship Modelling
http://blog.inf.ed.ac.uk/da16/tutorials

The exercise sheet for this tutorial will be on the course web pages. Tutorials are next week; start working on the exercises as soon as they are online.

As well as problems to do for the tutorial, these sheets contain several example questions with worked solutions.

/ January 26, 2016

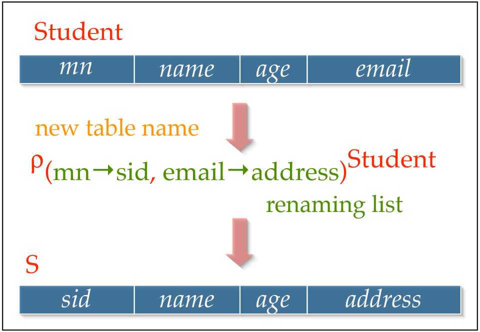
Bonus Off-Syllabus Slide +

Logical Operators

Truth	TRUE	$\top$	T	tt	1			
Falsity	FALSE	$\bot$	F	ff	0			
Conjunction	AND	$P \wedge Q$	&	&&	$*$	$\cap$	$\bullet$	.
Disjunction	OR	$P \vee Q$	|	||	$+$	$\cup$		
Implication	IMPLIES	$P \Rightarrow Q$			$\rightarrow$	$\supset$		
Equivalence	IFF	$P \Leftrightarrow Q$			$\leftrightarrow$	$\equiv$		
Negation	NOT	$\neg P$	!	~				

/ January 26, 2016

Renaming



Renaming changes the names of some or all fields in a table, giving a schema of the same arity and type.

This can be used to avoid *naming conflicts* when combining tables.

Union

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

S_1

mn	*name*	*age*	*email*
s0489967	Basil	19	basil@inf
s0412375	Mary	18	mary@inf
s9989232	Ophelia	24	oph@bio
s0189034	Peter	22	peter@math
s0289125	Michael	21	mike@geo

S_2

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math
s0489967	Basil	19	basil@inf
s9989232	Ophelia	24	oph@bio
s0289125	Michael	21	mike@geo

$S_1 \cup S_2$

Union combines the rows of two tables that have the same schema.

Difference

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

S_1

mn	*name*	*age*	*email*
s0489967	Basil	19	basil@inf
s0412375	Mary	18	mary@inf
s9989232	Ophelia	24	oph@bio
s0189034	Peter	22	peter@math
s0289125	Michael	21	mike@geo

S_2

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0378435	Helen	20	helen@phys

S_1-S_2

Difference takes all the rows of one table which do not appear in another.

Intersection

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

S_1

mn	*name*	*age*	*email*
s0489967	Basil	19	basil@inf
s0412375	Mary	18	mary@inf
s9989232	Ophelia	24	oph@bio
s0189034	Peter	22	peter@math
s0289125	Michael	21	mike@geo

S_2

mn	*name*	*age*	*email*
s0412375	Mary	18	mary@inf
s0189034	Peter	22	peter@math

$S_1 \cap S_2$

Intersection takes all the rows of one table which do appear in another.

$$S_1 \cap S_2 = S_1 - (S_1 - S_2)$$

Definitions

Union

Relation $R_1 \cup R_2$ contains every tuple that appears in either R_1 or R_2.

Difference

Relation $R_1 - R_2$ contains every tuple that appears R_1 but not in R_2.

Intersection

Relation $R_1 \cap R_2$ contains every tuple that appears in R_1 and also in R_2.

In all of these cases the schemas of R_1 and R_2 must be *compatible* — all the same fields with all the same types.

Intersection can be defined in terms of difference, but not the other way around. (Try it and see)

Cross Product

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

S

code	*name*	*year*
inf1	Informatics 1	1
math1	Mathematics 1	1

R

mn	*name*	*age*	*email*	*code*	*name*	*year*
s0456782	John	18	john@inf	inf1	Informatics 1	1
s0456782	John	18	john@inf	math1	Mathematics 1	1
s0412375	Mary	18	mary@inf	inf1	Informatics 1	1
s0412375	Mary	18	mary@inf	math1	Mathematics 1	1
s0378435	Helen	20	helen@phys	inf1	Informatics 1	1
s0378435	Helen	20	helen@phys	math1	Mathematics 1	1
s0189034	Peter	22	peter@math	inf1	Informatics 1	1
s0189034	Peter	22	peter@math	math1	Mathematics 1	1

$S \times R$

Cross product combines every row of one table with every row of another.

Definition

Cross product

For any relations R and S, the *cross product* $R \times S$, also known as the *Cartesian product*, is a relation defined as follows.

Schema

All the fields and types from R, plus all fields and types from S.
If necessary the renaming operation ρ can ensure none of these clash.

Rows

For every row $(u_1, \ldots, u_n)$ of R and every row $(v_1, \ldots, v_m)$ of S the product $R \times S$ contains row $(u_1, \ldots, u_n, v_1, \ldots, v_m)$.

The arity of $R \times S$ is the sum of the arities of R and S.

The cardinality of $R \times S$ is the product of the cardinalities of R and S.

Relational Join

The most commonly used relational operation is the *join* $R \bowtie_P S$ which combines cross-product with selection.

Rows in Join

For every row $(u_1, \ldots, u_n)$ of R and every row $(v_1, \ldots, v_m)$ of S the join relation $R \bowtie_P S$ contains row $(u_1, \ldots, u_n, v_1, \ldots, v_m)$ if and only if that tuple of values satisfies predicate P.

Here R and S are any two relations, with P any predicate defined on the fields of R and S together

$$R \bowtie_P S = \sigma_P(R \times S)$$

Example of Join

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Student

mn	*code*	*mark*
s0412375	inf1	80
s0378435	math1	70

Takes

mn	*name*	*age*	*email*	*mn*	*code*	*mark*
s0456782	John	18	john@inf	s0412375	inf1	80
s0456782	John	18	john@inf	s0378435	math1	70
s0412375	Mary	18	mary@inf	s0412375	inf1	80
s0412375	Mary	18	mary@inf	s0378435	math1	70
s0378435	Helen	20	helen@phys	s0412375	inf1	80
s0378435	Helen	20	helen@phys	s0378435	math1	70
s0189034	Peter	22	peter@math	s0412375	inf1	80
s0189034	Peter	22	peter@math	s0378435	math1	70

$$\sigma_{Student.mn = Takes.mn}(Student \times Takes)$$

Example of Join

mn	*name*	*age*	*email*
s0456782	John	18	john@inf
s0412375	Mary	18	mary@inf
s0378435	Helen	20	helen@phys
s0189034	Peter	22	peter@math

Student

mn	*code*	*mark*
s0412375	inf1	80
s0378435	math1	70

Takes

mn	*name*	*age*	*email*	*mn*	*code*	*mark*
s0456782	John	18	john@inf	s0412375	inf1	80
s0456782	John	18	john@inf	s0378435	math1	70
s0412375	Mary	18	mary@inf	s0412375	inf1	80
s0412375	Mary	18	mary@inf	s0378435	math1	70
s0378435	Helen	20	helen@phys	s0412375	inf1	80
s0378435	Helen	20	helen@phys	s0378435	math1	70
s0189034	Peter	22	peter@math	s0412375	inf1	80
s0189034	Peter	22	peter@math	s0378435	math1	70

$$\textit{Student} \bowtie_{\textit{Student.mn} = \textit{Takes.mn}} \textit{Takes}$$

Refined Joins

In general, a join $R \bowtie_P S$ can use an arbitrary predicate P.

However, some kinds of predicate are particularly common, and often followed by projection to eliminate duplicate or redundant columns.

Equijoin

An *equijoin* starts with a join where the predicate states that particular fields from each relation must be equal.

That is, P has the form $(a_1 = b_1) \wedge \cdots \wedge (a_k = b_k)$ for some fields $a_1, \ldots a_k$ of R and $b_1, \ldots, b_k$ of S.

For example, the relation $(\mathsf{Student} \bowtie_{\mathsf{Student.mn}=\mathsf{Takes.mn}} \mathsf{Takes})$ above is an equijoin between these tables on the two mn fields.

It's common to then project onto only certain columns to remove the fields that are now duplicated.

Natural Join

The *natural join* $R \bowtie S$ of relations R and S is the equijoin requiring equalities between any fields in the two relations that share the same name, followed by a projection to remove duplicate columns.

For example, the natural join of the "Student" and "Takes" relations:

$$\text{Student} \bowtie \text{Takes} =$$
$$\pi_{\substack{\text{mn,name,age,}\\ \text{email,code,mark}}}(\sigma_{\text{Student.mn}=\text{Takes.mn}}(\text{Student} \times \text{Takes}))$$

This records every student in combination with every course they take.

This example is typical: a natural join between two tables where one has a foreign key constraint referring to the other.

Specialist Joins

The SQL standard defines no less than five different types of join.

Inner Join is the basic join $R \bowtie_P S$ described earlier.

Left Outer Join is the basic join, plus rows for every tuple in the left-hand table R that matches nothing in the right-hand table S. Missing fields are filled with **NULL**.

Right Outer Join is the basic join plus rows for every tuple in the right-hand table S that matches nothing in the left-hand table R. Missing fields are filled with **NULL**.

Full Outer Join has every row from all three previous joins.

Cross Join is the cross-product $R \times S$, with every tuple from R paired with every tuple from S, and no matching done at all.