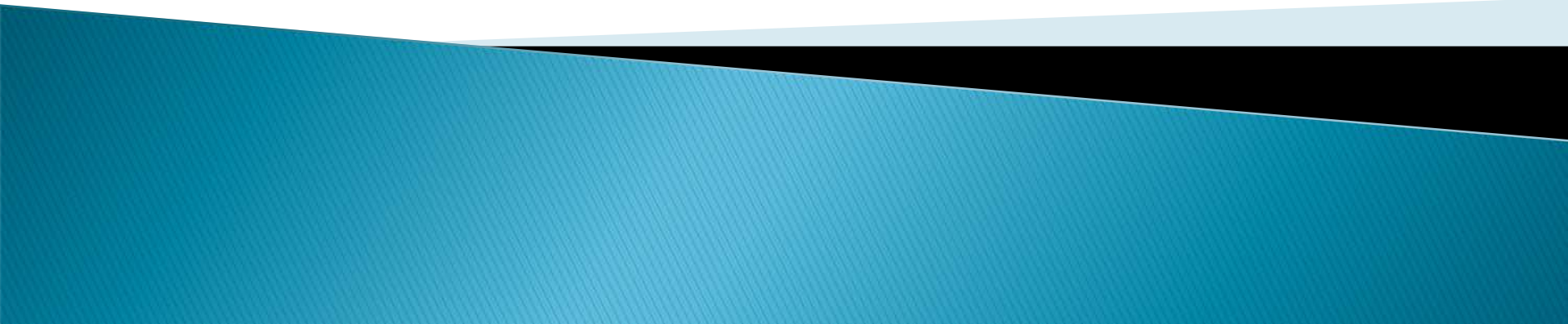


CMSC424: Database Design

Relational Model



Relational Data Model


Introduced by Ted Codd (late 60's – early 70's)

- *Before = “Network Data Model” and “Hierarchical Data Model” (e.g. IMS)*
- *Very contentious: Database Wars (Charlie Bachman vs. Ted Codd)*

Relational data model contributes:

1. *Separation of logical, physical data models (data independence)*
2. *Declarative query languages*
3. *Formal semantics*
4. *Query optimization (key to commercial success)*

1st prototypes:

- *Ingres → CA → Actian*
 - *Postgres → Illustra → Informix → IBM*
 - *System R → Oracle, DB2*
- 

Key Abstraction: Relation

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Terms:

- Tables (aka: Relations)

Why called Relations?

*Closely correspond to mathematical concept of a **relation***

Relations

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Considered equivalent to...

$\{ (Downtown, A-101, 500), \\ (Brighton, A-201, 900), \\ (Brighton, A-217, 500) \}$

Relational database semantics defined in terms of mathematical relations

Relations

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Considered equivalent to...

*{ (Downtown, A-101, 500),
(Brighton, A-201, 900),
(Brighton, A-217, 500) }*

Terms:

- Tables (aka: Relations)
- Rows (aka: tuples)
- Columns (aka: attributes)
- Schema (e.g.: Acct_Schema = (bname, acct_no, balance))

Definitions

Relation Schema (or Schema)

A list of attributes and their domains

*E.g. **account**(account-number, branch-name, balance)*

Programming language equivalent: A variable (e.g. x)

Relation Instance

A particular instantiation of a relation with actual values

Will change with time

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Programming language equivalent: Value of a variable

Definitions

Domains of an attribute/column

The set of permitted values

e.g., bname must be String, balance must be a positive real number

We typically assume domains are **atomic**, i.e., the values are treated as indivisible (specifically: you can't store lists or arrays in them)

Null value

A special value used if the value of an attribute for a row is:

unknown (e.g., don't know address of a customer)

inapplicable (e.g., "spouse name" attribute for a customer)

withheld/hidden

Different interpretations all captured by a single concept – leads to major headaches and problems

Tables in a University Database

classroom(building, room_number, capacity)

department(dept_name, building, budget)

course(course_id, title, dept_name, credits)

instructor(ID, name, dept_name, salary)

section(course_id, sec_id, semester, year, building,
room_number, time_slot_id)

teaches(ID, course_id, sec_id, semester, year)

student(ID, name, dept_name, tot_cred)

takes(Id, course_id, sec_id, semester, year, grade)

advisor(s_ID, i_ID)

time_slot(time_slot_id, day, start_time, end_time)

prereq(course_id, prereq_id)



Keys

- ▶ Let $K \subseteq R$ (set of all attributes)
- ▶ K is a **superkey** of R if values for K are sufficient to identify a unique tuple of any possible relation $r(R)$
 - *Example: $\{ID\}$ and $\{ID, name\}$ are both superkeys of instructor.*
- ▶ Superkey K is a **candidate key** if K is **minimal** (i.e., no subset of it is a superkey)
 - *Example: $\{ID\}$ is a candidate key for Instructor*
- ▶ One of the candidate keys is selected to be the **primary key**
 - Typically one that is small and immutable (doesn't change often)
- ▶ Primary key typically highlighted (e.g., underlined)

Tables in a University Database

course(course_id, title, dept_name, credits)

instructor(ID, name, dept_name, salary)



Tables in a University Database

takes(ID, course_id, sec_id, semester, year, grade)

What about ID, course_id?

No. May repeat:

("1011049", "CMSC424", "101", "Spring", 2014, D)

("1011049", "CMSC424", "102", "Fall", 2015, null)

What about ID, course_id, sec_id?

May repeat:

("1011049", "CMSC424", "101", "Spring", 2014, D)

("1011049", "CMSC424", "101", "Fall", 2015, null)

What about ID, course_id, sec_id, semester?

Still no: ("1011049", "CMSC424", "101", "Spring", 2014, D)

("1011049", "CMSC424", "101", "Spring", 2015, null)

Tables in a University Database

classroom(**building, room_number**, capacity)

department(**dept_name**, building, budget)

course(**course_id**, title, dept_name, credits)

instructor(**ID**, name, dept_name, salary)

section(**course_id, sec_id, semester, year**, building,
room_number, time_slot_id)

teaches(**ID, course_id, sec_id, semester, year**)

student(**ID**, name, dept_name, tot_cred)

takes(**ID, course_id, sec_id, semester, year**, grade)

advisor(**s_ID**, i_ID)

time_slot(**time_slot_id, day, start_time**, end_time)

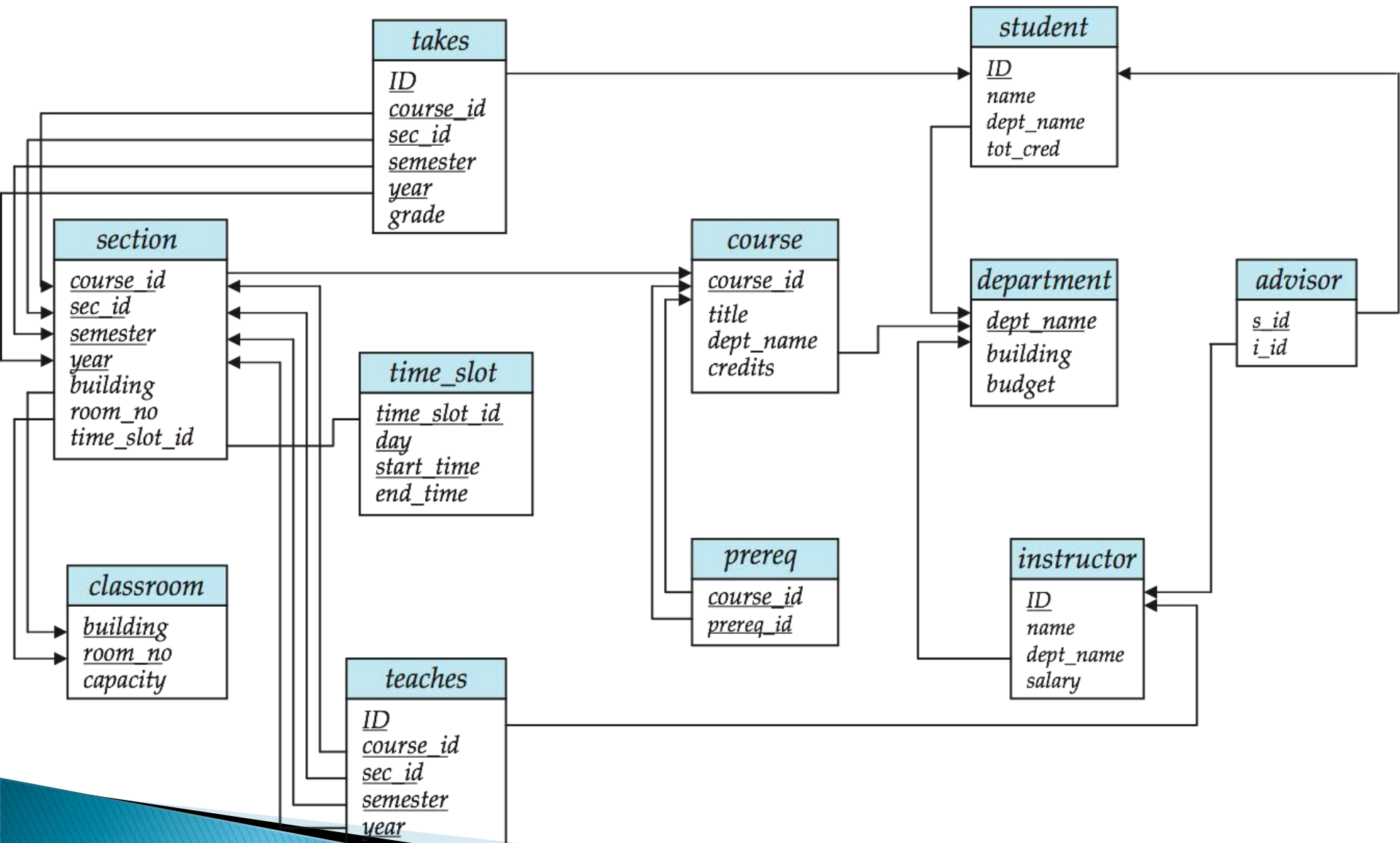
prereq(**course_id, prereq_id**)



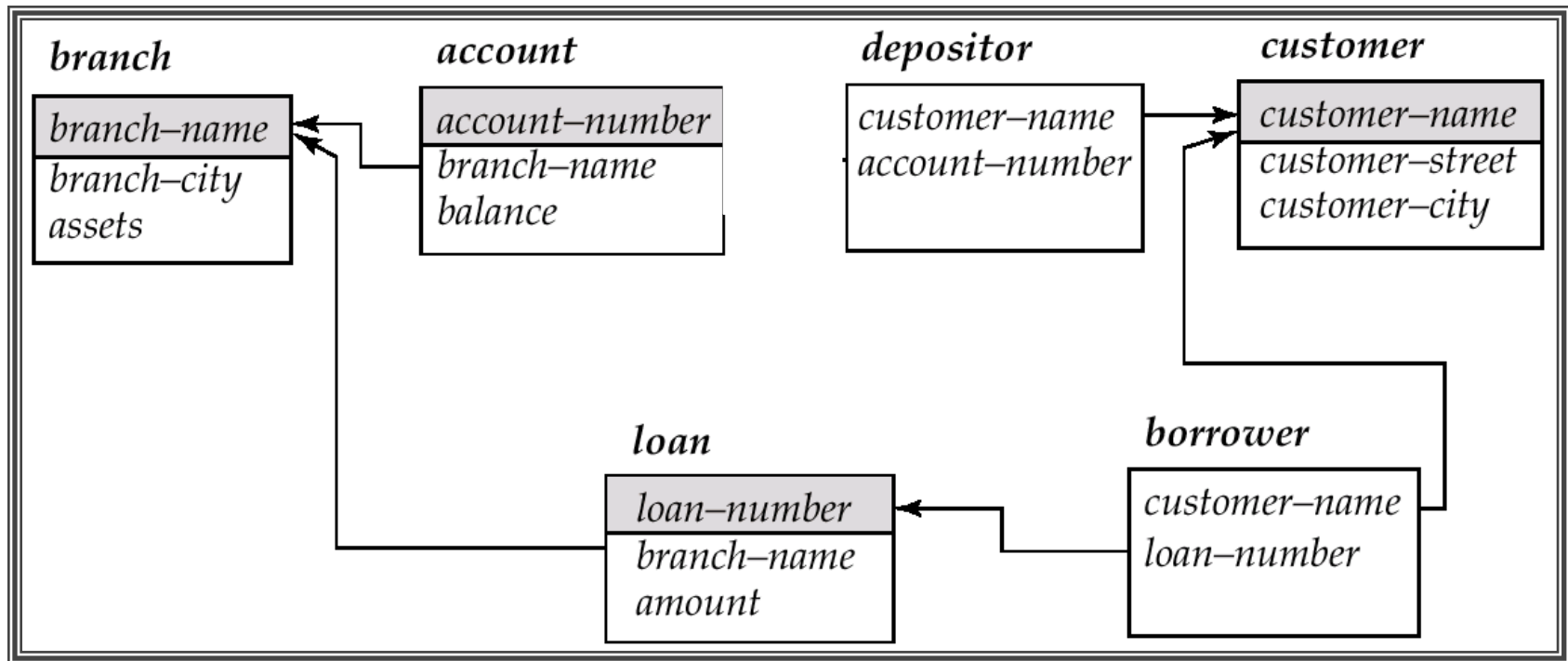
Keys

- ▶ **Foreign key:** Primary key of a relation that appears in another relation
 - {ID} from *student* appears in *takes*, *advisor*
 - *student* called **referenced** relation
 - *takes* is the **referencing** relation
 - Typically shown by an arrow from referencing to referenced
- ▶ **Foreign key constraint:** the tuple corresponding to that primary key must exist
 - Imagine:
 - Tuple: ('student101', 'CMSC424') in *takes*
 - But no tuple corresponding to 'student101' in *student*
 - Also called **referential integrity constraint**

Schema Diagram for University Database



Schema Diagram for the Banking Enterprise



Examples

- ▶ Account(cust_ssn, account_number, cust_name, balance, cust_address)
- ▶ RA(student_id, project_id, supervisor_id, appt_start_date, appt_end_date)
- ▶ Person(Name, DOB, Born, Education, Number of children, ...)
 - *Information typically found on Wikipedia Pages*

Examples

- ▶ Account(cust_ssn, account_number, cust_name, balance, cust_address)
 - If a single account per customer, then: cust_ssn
 - Else: (cust_ssn, account_number)
 - In the latter case, this is not a good schema because it requires repeating information
- ▶ RA(student_id, project_id, supervisor_id, appt_start_date, appt_end_date)
 - Could be smaller if there are some restrictions – requires some domain knowledge of the data being stored
- ▶ Person(Name, DOB, Born, Education, Number of children, ...)
 - *Information typically found on Wikipedia Pages*
 - *Unclear what could be a primary key here: you could in theory have two people who match on all of those*

Relational Query Languages

- ▶ Example schema: $R(A, B)$
- ▶ Practical languages
 - SQL
 - select A from R where B = 5;
 - Datalog (sort of practical)
 - $q(A) :- R(A, 5)$
- ▶ Formal languages
 - Relational algebra
$$\pi_A (\sigma_{B=5} (R))$$
 - Tuple relational calculus
$$\{ t : \{A\} \mid \exists s : \{A, B\} (R(A, B) \wedge s.B = 5) \}$$
 - Domain relational calculus
 - Similar to tuple relational calculus

Relational Operations

- ▶ Some of the languages are “procedural” and provide a set of operations
 - Each operation takes one or two relations as input, and produces a single relation as output
 - Example: Relational Algebra
- ▶ The “non-procedural” (also called “declarative”) languages specify the output, but don’t specify the operations
 - Relational calculus
 - Datalog (used as an intermediate layer in quite a few systems today)

Select Operation

Choose a subset of the tuples that satisfies some predicate
Denoted by σ in relational algebra

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project

Choose a subset of the columns (for all rows)

Denoted by Π in relational algebra

Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$\Pi_{A,D}(r)$

A	D
α	7
α	7
β	3
β	10

A	D
α	7
β	3
β	10

Relational algebra following “set” semantics – so no duplicates
SQL allows for duplicates – we will cover the formal semantics later

Set Union, Difference

Relation r, s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

$r - s$:

A	B
α	1
β	1

Must be compatible schemas

What about intersection ?

Can be derived

How to express intersection?

$$r \cap s =$$

- A. $r - (r \cup s);$
- B. $r - (r - s);$
- C. $r \cup (r \cup s);$
- D. $r \cup (r \cap s);$

Cartesian Product

Combine tuples from two relations

If one relation contains N tuples and the other contains M tuples, the result would contain N*M tuples

The result is rarely useful – almost always you want pairs of tuples that satisfy some condition

Relation r, s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Joins

Combine tuples from two relations if the pair of tuples satisfies some constraint

Equivalent to Cartesian Product followed by a Select

Relation r, s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

$r \bowtie_{A=C} s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Natural Join

Combine tuples from two relations if the pair of tuples agree on the common columns (with the same name)

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Figure 2.5 The *department* relation.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Figure 2.4 Unsorted display of the *instructor* relation.

department ⋈ instructor:

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

Figure 2.12 Result of natural join of the *instructor* and *department* relations.