Relational Model

- A *relation scheme* is a finite sequence of unique attribute names. For example,

  `EMPLOYEES = (EMPID, ENAME, ADDRESS, SALARY)`

  is a relation scheme with four attribute names.

- A *domain* is a set of values. With each attribute name, `A`, a domain, `dom(A)`, is associated. This domain includes a special value called *null*. For example, `dom(EMPID)` could be the set of all possible integers between 1000 and 9999 and the special null value.

- Given a relation scheme `R = A1, ..., An`, a *relation* `r` on the scheme `R` is defined as any finite subset of the Cartesian product

  `dom(A1) x ...  x dom(An)`.

- Assuming appropriate domains for the `EMPLOYEES` relation scheme, a sample relation under this scheme could be

```
{ (1111,'Jones','111 Ash St.',20000),
  (2222,'Smith','123 Elm St.',25000),
  (3333,'Brown','234 Oak St.',30000) }
```

- Each of the elements of a relation is also referred to as a *tuple*.
- A *relational database scheme*, `D`, is a finite set of relation schemes,

  { `R1, ..., Rm` }.

- A *relational database* on scheme `D` is a set of relations

  { `r1, ...  rm` }

  where each `ri` is a relation on the corresponding scheme `Ri`.

Key Constraint

- A *super key* for a relation scheme `R` is any subset, `K` of `R` that satisfies the property that in every valid relation under the scheme `R`, it is not possible to have two different tuples with the same values under `K`.
- A *candidate key* for `R` is any key for `R` such that none of its proper subsets is also a key.
- The *primary key* for a relation scheme `R` is one of the candidate keys chosen by the designer of the database.
- The primary key attributes are required to satisfy the **`not null`** constraint, i.e., no tuple can have a **`null`** value under the primary key attributes.

Referential Integrity/Foreign Key Constraint

- During the design of a relational database, the designer may create a relation scheme `R` which includes the primary key attributes of another relation scheme, say `S`.
- In such a situation, the referential integrity constraint specifies the condition that the values that appear under the primary key attributes in any valid relation under scheme `R` **must** also appear in the relation under scheme `S`.
- The attributes in the scheme `R` that correspond to the primary key attributes of scheme `S` collectively are referred to as a *foreign key* in scheme `R`.
- Unlike the primary key attributes, the foreign key attributes do not have to satisfy the `not null` constraint.

Not Null constraint

- This constraint specifies the condition that tuple values under certain attributes (specified to be not null) cannot be null.
- This condition is usually always imposed on the primary key attributes. In Oracle, primary key attributes are automatically constrained to be **`not null`**.
- Other attributes may also be constrained to be not null if the need arises.

Grade book database

```
CATALOG(CNO,ctitle)
STUDENTS(SID,fname,lname,minit)
COURSES(TERM,LINENO,cno,a,b,c,d)
COMPONENTS(TERM,LINENO,COMPNAME,maxpoints,weight)
ENROLLS(SID,TERM,LINENO)
SCORES(SID,TERM,LINENO,COMPNAME,points)
```

catalog

CNO	CTITLE
csc226	Introduction to Programming I
csc227	Introduction to Programming II
csc343	Assembly Programming
csc481	Automata and Formal Languages
csc498	Introduction to Database Systems
csc880	Deductive Databases and Logic Programming

students

SID	FNAME	LNAME	MINIT
1111	Nandita	Rajshekhar	K
2222	Sydney	Corn	A
3333	Susan	Williams	B
4444	Naveen	Rajshekhar	B
5555	Elad	Yam	G
6666	Lincoln	Herring	F

courses

TERM	LINENO	CNO	A	B	C	D
f96	1031	csc226	90	80	65	50
f96	1032	csc226	90	80	65	50
sp97	1031	csc227	90	80	65	50

components

TERM	LINENO	COMPNAME	MAXPOINTS	WEIGHT
f96	1031	exam1	100	30
f96	1031	quizzes	80	20
f96	1031	final	100	50
f96	1032	programs	400	40
f96	1032	midterm	100	20
f96	1032	final	100	40
sp97	1031	paper	100	50
sp97	1031	project	100	50

enrolls

SID	TERM	LINENO
1111	f96	1031
2222	f96	1031
4444	f96	1031
1111	f96	1032
2222	f96	1032
3333	f96	1032
5555	sp97	1031
6666	sp97	1031

scores

SID	TERM	LINENO	COMPNAME	POINTS
1111	f96	1031	exam1	90
1111	f96	1031	quizzes	75
1111	f96	1031	final	95
2222	f96	1031	exam1	70
2222	f96	1031	quizzes	40
2222	f96	1031	final	82
4444	f96	1031	exam1	83
4444	f96	1031	quizzes	71
4444	f96	1031	final	74

Mail order database

```
EMPLOYEES(ENO,ename,zip,hdate)
PARTS(PNO,pname,qoh,price,level)
CUSTOMERS(CNO,cname,street,zip,phone)
ORDERS(ONO,CNO,ENO,received,shipped)
ODETAILS(ONO,PNO,qty)
ZIPCODES(ZIP,city)
```

employees

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12-DEC-95
1001	Smith	60606	01-JAN-92
1002	Brown	50302	01-SEP-94

parts

PNO	PNAME	QOH	PRICE	LEVEL
10506	Land Before Time I	200	19.99	20
10507	Land Before Time II	156	19.99	20
10508	Land Before Time III	190	19.99	20
10509	Land Before Time IV	60	19.99	20
10601	Sleeping Beauty	300	24.99	20
10701	When Harry Met Sally	120	19.99	30
10800	Dirty Harry	140	14.99	30
10900	Dr. Zhivago	100	24.99	30

customers

CNO	CNAME	STREET	ZIP	PHONE
1111	Charles	123 Main St.	67226	316-636-5555
2222	Bertram	237 Ash Avenue	67226	316-689-5555
3333	Barbara	111 Inwood St.	60606	316-111-1234

orders

ONO	CNO	ENO	RECEIVED	SHIPPED
1020	1111	1000	10-DEC-94	12-DEC-94
1021	1111	1000	12-JAN-95	15-JAN-95
1022	2222	1001	13-FEB-95	20-FEB-95
1023	3333	1000	20-JUN-97	null

odetails

ONO	PNO	QTY
1020	10506	1
1020	10507	1
1020	10508	2
1020	10509	3
1021	10601	4
1022	10601	1
1022	10701	1
1023	10800	1
1023	10900	1

zipcodes

ZIP	CITY
67226	Wichita
60606	Fort Dodge
50302	Kansas City
54444	Columbia
66002	Liberal
61111	Fort Hays

Relational Algebra - Set-theoretic operations

- Two relations are **union-compatible** if they have the same number of attributes and the domains of the corresponding attributes in the two relations are the same.
- Consider two relations r and s that are union-compatible.

  Union: $r \cup s = \{t | t \in r \text{ or } t \in s\}$.

  Difference: $r - s = \{t | t \in r \text{ and } t \notin s\}$

  Intersection: $r \cap s = \{t | t \in r \text{ and } t \in s\}$

- **Cartesian Product:** Let r and s be any two relations.

  $$r \times s = \{t1.t2 | t1 \in r \text{ and } t2 \in s\},$$

  where, $t1.t2$ is the concatenation of tuples $t1$ and $t2$ to form a larger tuple.

Example: Set-theoretic Operators

r

A	B
a	b
a	c
b	d

s

A	B
a	c
a	e

$r \cup s$

A	B
a	b
a	c
b	d
a	e

$r - s$

A	B
a	b
b	d

$r \times s$

r.A	r.B	s.A	s.B
a	b	a	c
a	b	a	e
a	c	a	c
a	c	a	e
b	d	a	c
b	d	a	e

$r \cap s$

A	B
a	c

Relation-theoretic operations

Rename: The rename operator takes as input a relation and returns the same relation as output, but under a different name. The symbolic notation for the rename operator is $\rho_s(r)$, where r is the input relation and s is the new name.

Select: Symbolically, the select operator is written as $\sigma_F(r)$, where F is the selection criterion and r is the input relation and is defined as follows:

$$\sigma_F(r) = \{t | t \in r \text{ and } t \text{ satisfies } F\}.$$

Project: Symbolically, the project operator is written as $\pi_A(r)$, where A is a sub-list of the attributes of r, and is defined as follows:

$$\pi_A(r) = \{t[A] | t \in r\}$$

where $t[A]$ is a tuple constructed from t by keeping the values that correspond to the attributes in A and discarding other values.

Relation-theoretic operations - Continued

Natural Join: Symbolically, the natural join is written as $r \bowtie s$, where r is a relation on scheme R and s is a relation on scheme S, and is defined as follows:

$$r \bowtie s = \{t | (\exists u \in r)(\exists v \in s)(t[R] = u \text{ and } t[S] = v)\}$$

Division: Symbolically, the division operation is written as $r \div s$ and is defined as follows:

$$r \div s = \{t | (\forall u \in s)(t.u \in r)\}$$

where $t.u$ is the concatenation of tuple t with tuple u.

Example: Relation-theoretic operations

r

A	C	D
a	c	d
a	e	f
a	g	h
b	c	d
b	g	h
c	c	d
c	e	f

s

C	D
c	d
e	f

t

B	C	D
b	c	d
b	e	f

$\sigma_{A='b' \text{ or } C='c'}(r)$

A	C	D
a	c	d
b	c	d
b	g	h
c	c	d

$\pi_A(r)$

A
a
b
c

$r \bowtie t$

A	C	D	B
a	c	d	b
a	e	f	b
b	c	d	b
c	c	d	b
c	e	f	b

$r \div s$

A
a
c

Basic Operations

Basic set of operations: rename, select, project, cartesian product, union, difference. Other operations can be expressed in terms of these six.

Intersection: $r \cap s = r - (r - s)$

Natural Join: $r \bowtie s = \pi_{R \cap S}(\sigma_F(r \times s))$

where F is a selection condition which indicates that the tuple values under the common attributes of r and s are equal.

Division: $r \div s = \pi_{R-S}(r) - \pi_{R-S}((\pi_{R-S}(r) \times s) - r)$

Even though relation schemes are defined as sequences, they are treated as sets in these equalities for simplicity.

Querying using relational algebra

Gradebook database queries

Q1 Get the names of students enrolled in the `Assembly Programming` class in the `f96` term.

$$t1 := \sigma_{CTITLE='\text{Assembly Programming}'}(catalog)$$
$$t2 := \sigma_{TERM='f96'}(courses)$$
$$t3 := t1 \bowtie t2 \bowtie enrolls \bowtie students$$
$$result := \pi_{FNAME,LNAME,MINIT}(t3)$$

Q2 Get the `SID` values of students who did not enroll in any class during the `f96` term.

$$\pi_{SID}(students) - \pi_{SID}(\sigma_{TERM='f96'}(enrolls))$$

Gradebook database queries continued

Q3 Get the `SID` values of students who have enrolled in `csc226` and `csc227`,

$$t1 := \pi_{SID}(enrolls \bowtie \sigma_{CNO='csc226'}(courses))$$
$$t2 := \pi_{SID}(enrolls \bowtie \sigma_{CNO='csc227'}(courses))$$
$$result := t1 \cap t2$$

Q4 Get the `SID` values of students who have enrolled in `csc226` or `csc227`,

$$t1 := \pi_{SID}(enrolls \bowtie \sigma_{CNO='csc226'}(courses))$$
$$t2 := \pi_{SID}(enrolls \bowtie \sigma_{CNO='csc227'}(courses))$$
$$result := t1 \cup t2$$

Q5 Get the `SID` values of students who have enrolled in **all** the courses in the catalog.

$$\pi_{SID,CNO}(courses \bowtie enrolls) \div \pi_{CNO}(catalog)$$

Mail order database queries

Q6 Get part names of parts that cost less than `20.00`.

$$\pi_{PNAME}(\sigma_{PRICE<20.00}(parts))$$

Q7 Get pairs of CNO values of customers who have the the same zipcode.

$$t1 := \rho_{c1}(customers) \times \rho_{c2}(customers)$$
$$t2 := \sigma_{c1.ZIP=c2.ZIP \text{ and } c1.CNO<c2.CNO}(t1)$$
$$result := \pi_{c1.CNO,c2.CNO}(t2)$$

Q8 Get the names of customers who have ordered parts from employees living in `Wichita`.

$$t1 := \pi_{ENO}(employees \bowtie \sigma_{CITY='Wichita'}(zipcodes))$$
$$result := \pi_{CNAME}(customers \bowtie orders \bowtie t1)$$

Mailorder database queries continued

Q9 Get CNO values of customers who have ordered parts only from employees living in `Wichita`.

$$t1 := \pi_{ENO}(employees \bowtie \sigma_{CITY \neq 'Wichita'}(zipcodes))$$
$$result := \pi_{CNO}(orders) - \pi_{CNO}(orders \bowtie t1))$$

Q10 Get CNO values of customers who have ordered parts from all employees living in `Wichita`.

$$t1 := \pi_{ENO}(employees \bowtie \sigma_{CITY = 'Wichita'}(zipcodes))$$
$$result := \pi_{CNO,ENO}(orders) \div t1$$