3. The Relational Model

3.1 Relation Theory

Given a collection of sets $D_1, D_2, \ldots, D_n$ (not necessarily distinct), R is a relation on these n sets if it is a set of ordered n-tuples $\{A_1, A_2, \ldots\ldots, A_n\}$ such that

$A_1 \in D_1, A_2 \in D_2, \ldots, A_n \in D_n$

where $A_1, A_2, \ldots\ldots, A_n$ are the *attributes* of R
and the sets $D_1, D_2, \ldots, D_n$ are *domains* of R;

Relational Model (cont)

- a domain is the pool of values from which the attribute values actually appearing in any one column are drawn
  e.g. the domain of patient# could be an integer between 1000 and 9999
- the value of n is the *degree* of R
  (n = # of columns in the table/relation)

Relational schema

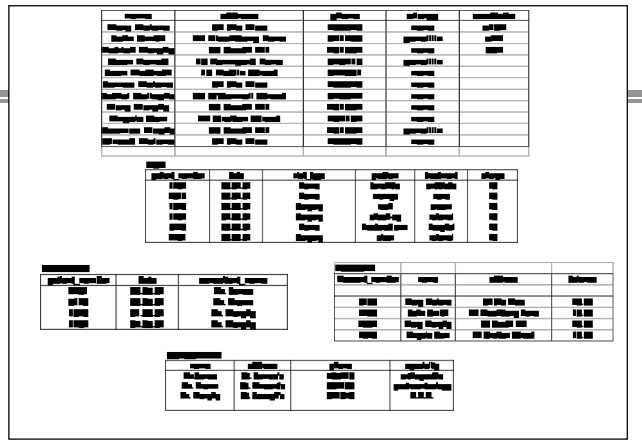
- relations are normally represented in quasi-mathematical form by a *relational schema* or *relation intension*:

  R (A_1, A_2, ..., A_n)

Example:

PATIENT (patient_number, name, address, phone, allergy)

Figure 1.2 shows the *relation extension*; i.e. it contains actual values



Null values

- it is possible to have null values (there are restrictions) in certain (row, column) positions in a relation if the value is not known or inapplicable
- the null value is a special value defined for each domain

3.2 Keys and integrity constraints

- a relational DB consists of many relations;
- every relation and every attribute has a name and hence can be uniquely identified
- attribute names are often qualified by the relation name to which they belong

e.g. PATIENT.name
CONSULTANT.name

Primary Keys

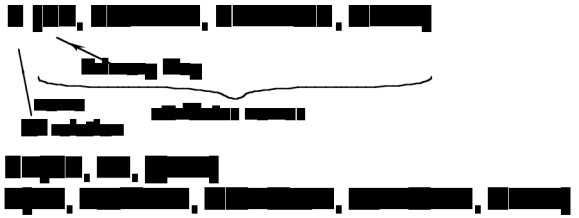
- most relations have one attribute where values uniquely identify the tuples of a relation

  e.g. patient_number in relation PATIENT - no two patients can have the same patient number;

  such a key is called a *primary key*

Primary Keys (cont)

- not every relation has a single attribute key e.g. S# and P# in SP



Integrity constraints

- ***semantic integrity constraints***

  relate to the meaning of attributes in the mini-world modelled by the DB e.g. a patient can only have one allergy

  entity constraints will be enforced automatically by the system

Integrity constraints (cont)

Structural integrity constraints

are part of relational model itself:

- *key constraints*, which are concerned with the uniqueness of primary keys
- *entity constraints*, which require that no part of a primary key is null
- once an attribute (possibly composite) is defined to be the primary key of a relation, key and entity constraints will be enforced automatically by the system

Integrity constraints (cont 2)

Foreign keys and referential integrity

referential integrity constraints are inter-relational constraints that govern relationships between relations

Example - A University DB

STUDENT(student_number, student_name, student_address)

RESULT (course_number, student_number, grade)

COURSE (course_number, course_title, lecturer)

University DB (cont)

- it would be logical to constrain values for student_number and course_number in the RESULT relation to be the same as existing tuples in the STUDENT and COURSE relations
- student_number and course_number in RESULT are referred to as foreign keys of the STUDENT and COURSE relations
- constraint on the values of student_number and course_number is called a referential integrity constraint

Foreign Keys (Formally)

A set of attributes FK in a relation schema R_1 is a foreign key of R_1 if it satisfies the following two rules:

1. The attributes FK have the same domain as the primary key attributes of another relation schema R_2; the attributes FK are said to reference or refer to the relation R_2.
2. A value of FK in a tuple t_1 of R_1 either occurs as a value of PK for some tuple t_2 in R_2 or is null. In the former case, we have $t_1[FK] = t_2[PK]$ and tuple t_1 references or refers to tuple t_2

3.3 Relational Algebra

- Traditional Data Schema Languages (DLS) operate on a tuple (or record) at a time e.g. hierarchial DB DSL
- Relational algebra provides us with a set-at-a-time language.
- Relational algebra is a collection of high-level operations on relations which fall into 2 groups:
  - traditional set operations (union, intersection, etc.)
  - relational operations (selection, projection, join, etc)
- All operations result in new relations

Set Operators

Traditional set operations

Union, Intersection, Difference

- For union, intersection and difference, 2 relations must be union compatible,

  i.e. they must be of the same degree (same number of columns!), n say, and the jth attribute of one must be drawn from the same domain as the jth attribute of the other

UNION

- The union of two (union-compatible) relations A and B,

  A UNION B

  is the set of all tuples t belonging to either A or B (or both).

Example Of Union

Let A be the set of supplier tuples for suppliers in London & B the set of supplier tuples for suppliers who supply part P1.

Then A UNION B

is the set of supplier tuples for suppliers who either are located in London or supply part P1 (or both).

INTERSECTION

The intersection of two (union-compatible) relations A and B,

A INTERSECT B

is the set of all tuples t belonging to both A and B

Example of Intersection

Example: Let A be the set of supplier tuples for suppliers in London, and B the set of supplier tuples for suppliers who supply part P1

Then A INTERSECT B

is the set of supplier tuples who are located in London and who supply part P1.

Difference

The difference between two (union-compatible) relations A and B,

A MINUS B

is the set of all tuples belonging to A and not to B.

Example Of Difference

Let A be the set of supplier tuples for suppliers in Paris and B the set of supplier tuples for suppliers who supply part P1

Then A MINUS B

is the set of supplier tuples for suppliers who are located in Paris and who do not supply part P1.

Relational Operators

SPECIAL RELATIONAL OPERATORS

We will look at 3 operators:

(a) ***selection***

(b) ***projection***

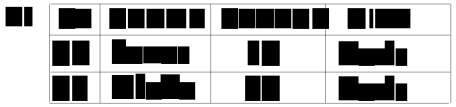
(c) ***join***

SELECTION

- The SELECT operation is an operation for constructing a horizontal subset of an existing relation,
  i.e. all rows (tuples) of an existing relation which can satisfy some conditions.

Examples:

SELECT S WHERE CITY='PARIS' GIVING T1;



Examples of Selection

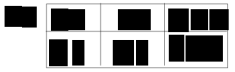
SELECT P WHERE WEIGHT < 17 GIVING T2;

T2

P#	Pname	Colour	Weight	City
P1	Nut	Red	12	London
P4	Screw	Red	14	London

Example of SELECTION (2)

SELECT SP WHERE S#='S1' AND P#='P1' GIVING T3;



PROJECTION

PROJECT is an operator for constructing a vertical subset of a relation

i.e. a subset obtained by selecting specified attributes and eliminating others, (also eliminating duplicate tuples within the attributes selected).

Example Of Projection

PROJECT S OVER CITY GIVING T4;

CITY
London
Paris

PROJECT S OVER SNAME, CITY, S#, STATUS GIVING T5;

T5

SName	City	Status	S#
Smith	London	20	S1
Jones	Paris	10	S2
Blake	Paris	30	S3

JOIN

- If two relations have a domain in common they may be joined over that domain;
- the result of the join is a new relation of higher degree in which each row is formed by joining together two rows;
- one from each of the original tables such that the two rows concerned have the same value in the common domain.

Example Joins

JOIN S AND P OVER CITY GIVING T6;

T6

S#	SName	Status	City	P#	Pname	Colour	Weight
S1	Smith	20	London	P1	Nut	Red	12
S1	Smith	20	London	P4	Screw	Red	14
S2	Jones	10	Paris	P2	Bolt	Green	17
S3	Blake	30	Paris	P2	Bolt	Green	17

A join in which one of the two identical columns is eliminated is called a natural join.