Data Model

*What is a Data Model?

* Data Abstraction: A data model is used to hide storage details and present the users with a conceptual view of the database.
* Data Model: A set of concepts to describe the structure of a database, and certain constraints that the database should obey.

- To represent the real world information (physical and logic objects) in a well formulated format such that it is easy for machine processing.

- A set of structures and operator which allows the description of data from both the static and dynamic point of view.
- Example. Student(name, ID, course, grade).

* Components of a Data Model
  1) Structures: basic categories in which data are organized. (defined by DEA'S)
  2) Operations: actions that can be performed on the structures. (defined by DML)
  3) Integrity Constrains: restrictions on the structures.

  Examples: Relational, Hierarchical, Network, E-R models.

  (for record-oriented)

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Fig. The suppliers-and-parts database

Q1: Find supplier number for suppliers who supply part P2	Q2: Find part number for parts supplied by supplied by supplier S2

Do until no more shipments;
get next shipment
where P# = P2;
print S#;
end;
	Do until no more shipments;
	get next shipment
	where S# = S2
	print P#
	end;

Fig. The Relation View

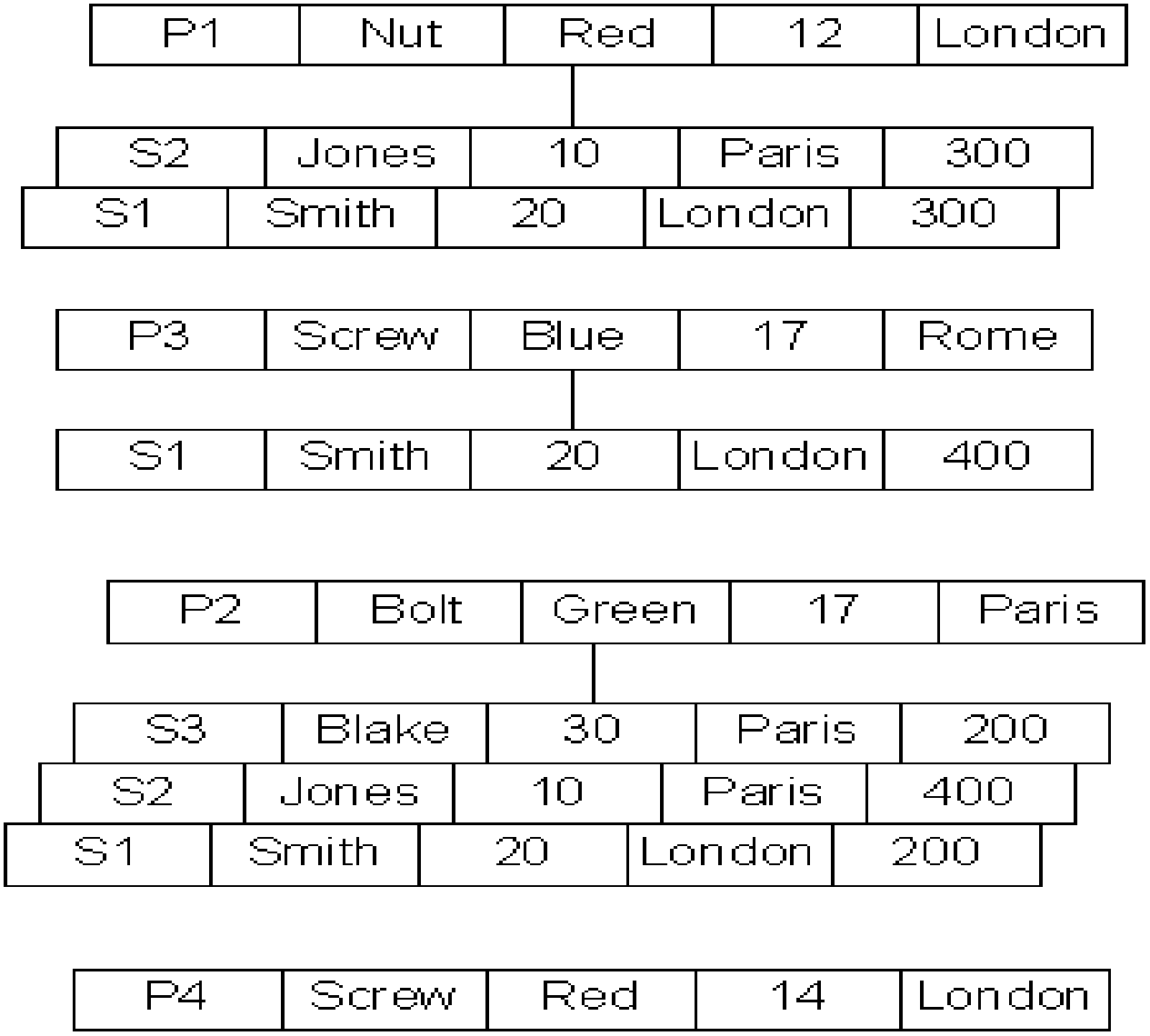
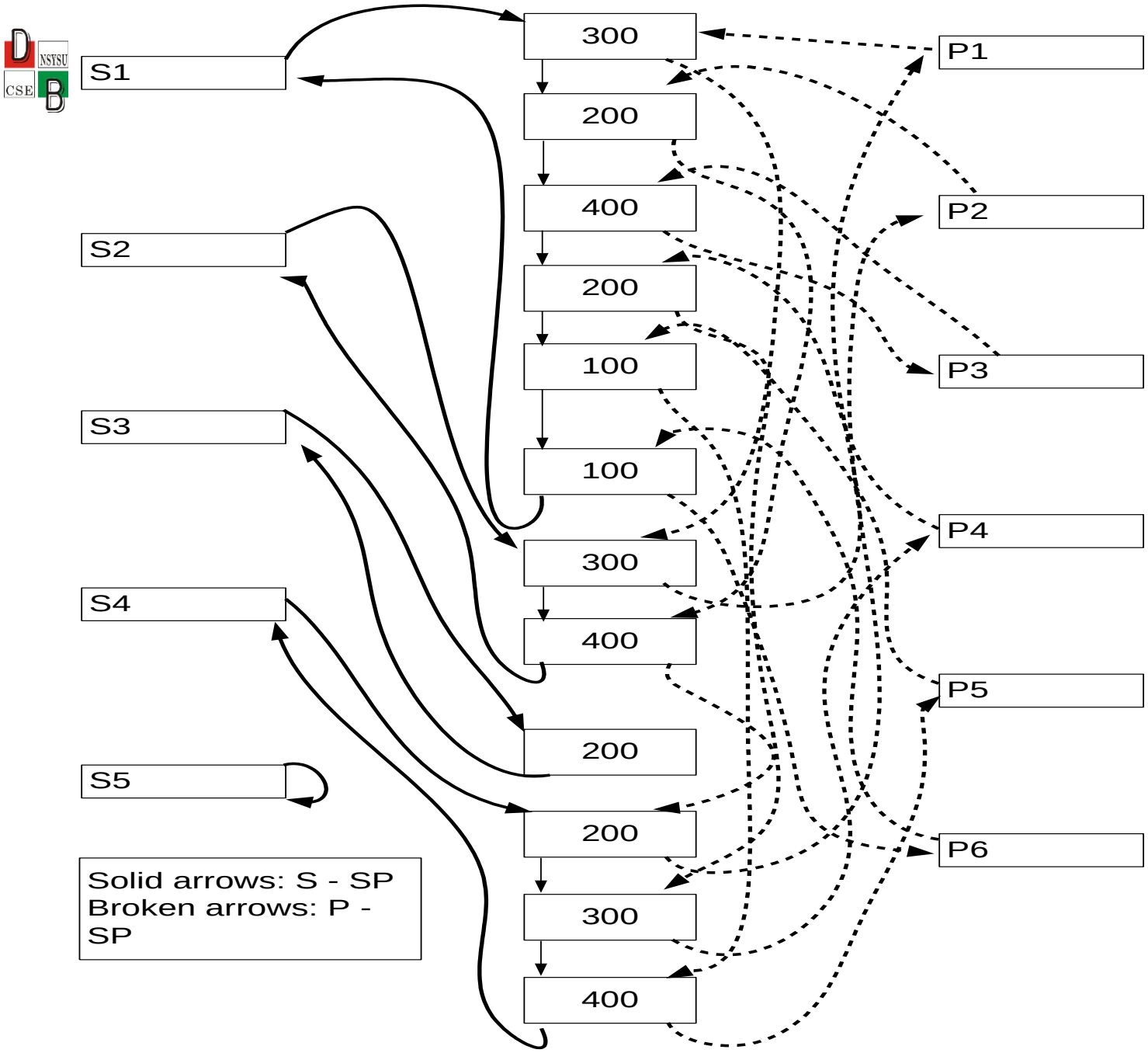


Fig. Hierarchical Form

Q1: Find supplier number for suppliers who supply part P2	Q2: Find part number for parts supplied by supplied by supplier S2

get [next] part where P# = P2;
do until no more suppliers
under this part;
get next supplier
under this part;
print S#
end;
	do until no more part;
	get next part;
	get [next] supplier
	under this part
	where S# = S2;
	if found then print P#;
	end;

Fig. The Hierarchical View



The Network View

Q1: Find supplier number for suppliers who supply part P2	Q2: Find part number for parts supplied by supplied by supplier S2

get [next] part where P# = P2;
do until no more connectors
under this part;
get next connector
under this part;
get supplier over this
connector;
print S#;
end;
	get [next] supplier where S# = S2;
	do until no more connectors
	under this supplier;
	get next connector
	under this supplier;
	get part over this
	connector;
	print P#;
	end;

Fig. The Network View

The Relational Database Model

* Relation (informally): A table of values. Each column in the table has a column header called an attribute. Each row is called a tuple.
* Formal Definition
* Domain: A pool of atomic (indivisible) values from which the actual values appearing in a given column are drawn.

- A set of scalar values, all of the same data type.
- Why domain ? domains constrain comparisons. (join, union)

* Attribute: A name to suggest the meaning that a domain plays in a particular relation. Each attribute Aj has a domain Dj.

* Relation Schema: R(A1, A2, ..., An)

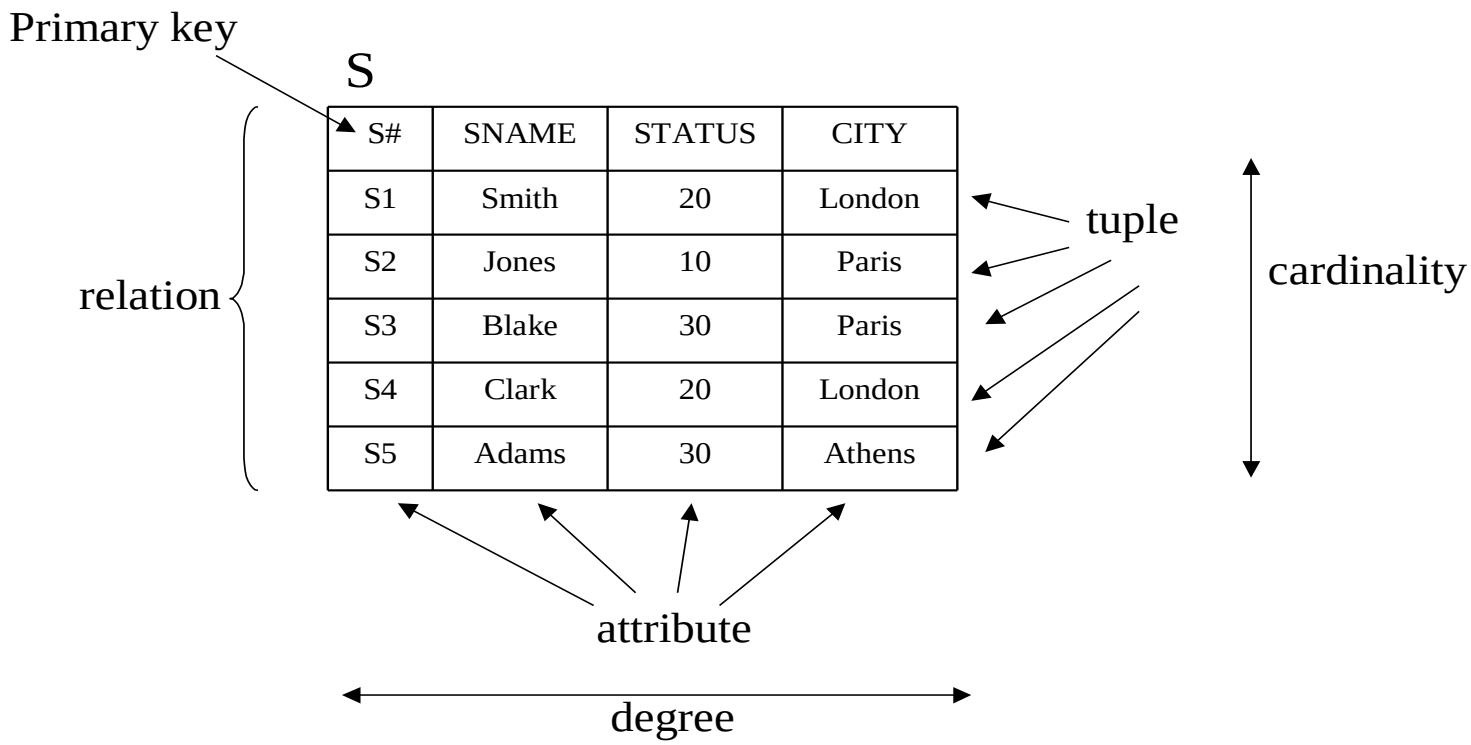


Fig. Relational Data Structure

Formal relational term	Informal equivalents
Relation	Table
Tuple	Row or record
Cardinality	Number of rows
Attribute	Column or field
Degree	Number of columns
Primary key	Unique identifier
Domain	Pool of legal values

Fig. Data structure terminology

S#	P#
S2	P1
S2	P2
S4	P2
S4	P4
S4	P5

Instead of

S#	P#
S2	{ P1, P2 }
S4	{ P2, P4, P5 }

Fig. Atomic values

before

S#	PQ	
	P#	QTY
S2	P1	300
	P2	400
S4	P2	200
	P4	400

after

S#	P#	**QTY**
S2	P1	300
S2	P2	400
S4	P2	200
S4	P4	400

Fig. An example of normalization

* A relation (R) on a collection of domains D1, D2, .. Dn (not necessary all distinct) consists of two parts, a heading and a body.

1) The heading consists of a fixed set of attributes, or more precisely attribute domain pairs, {(A1:D1), (A2:D2), ... (An:Dn)} such that each attribute Aj corresponds to exactly one of the underlying domains Dj (j= 1, 2, .., n).

2) The body consists of a time-varying set of tuples, where each tuple in turn consists of a set of attribute-value pairs, {(A1, vi1), (A2, vi2), .. (An, vin)} (i = 1,2,..,m, where m is the number of tuples in the set).

- In each such tuple, there is one such attribute-value pair (Aj, vij) for each attribute Aj in the heading. For any given attribute-value pair (Aj, vij), vij is a value from the unique domain Dj that is associated with the attribute Aj.
  (A mathematic set).

- (m: cardinality; n: degree)

* Given sets D1, D2, ..., Dn (not necessarily distinct), R is a relation on those n sets if it is a set of n-tuples each of which has its first element from D1, its second element from D2, and so on. R is a subset of the Cartesian Product
  D1 * D2 * ... * Dn of its domains.

*Properties of Relations

1) There are no duplicate tuples. (a set)
2) Tuples are unordered (top to bottom).(a set)
3) Attributes are unordered (left to right) (different from DB2). (referred by name, never by position)
4) All attribute values are atomic. (at every row-and column position within the table, there always exists precisely one value, never a list of values)

- Relations do not contain repreating groups.
- A relation satisfying this condition is said to be normalized.

- Reasons for normalization: a normalized relation is a simpler structure than an unnormalized one.
  As a result, the corresponding operators are simpler too.

5) A special null value is used to represent values that are unknown or inapplicable to certain tuples.

* A Relational database is a database that is perceived by the user as a collection of time-varying, normalized relations of assorted degrees.
  - A relation --> a file.
  - A tuple --> record.
  - An attribute --> a field.

* Kinds of relations: Base tables and views.

*Relational Integrity Constraints

* Constraints are conditions that must hold on all valid relation instances.
  Ex. shipment quantities must be a multiple of 100;
  Ex. all red parts must be stored in London;
* Three main types of constraints:
  Key, entity integrity, referential integrity constraints.

* Candidate Key: Attribute K (possible composite) of relation R is a candidate key for R if and only if it satisfies the following two time-independent properties:

1) Uniqueness: at any given time, no two tuples of R have the same value for K.

2) Minimality: if K is composite, then no component of K can be eliminated without destroying the uniqueness property.

* Every relation does have at least one candidate key, because relations do not contain duplicate tuples.

* Primary key: Choose one of those candidate keys to be primary keys, the remainder are alternative keys. A column or column combination with the property that, at any given time, no two rows of the table contain the same value in the column or column combination.

*The Entity Integrity Rule

"No component of the primary key of a base relation is allowed to accept nulls.“

- An entity without an identity does not exist.
- Entity integrity rule applies only to primary keys.
- Alternative keys may or may not have "nulls allowed."
- Primary key values must be unique. (basic definition)

- However, the entity integrity rule says that primary keys in base relations must not contain any nulls. (it is different from the definition of a primary key.)

*Foreign Key

A foreign key is an attribute (possible composite) of

relation R2 whose values are required to match those of

the primary key of some relation R1 (R1 and R2 not

necessarily distinct).

- In example SP (S#, P#, QTY), a given value for that attribute should be permitted to appear in the database only if that same value also appears as a value of the primary key S# of relation S.(the converse is not a requirement).

- A foreign key value (SP.S#, SP.P#) represents a reference to the tuple containing the matching primary key value (the referenced tuple of target tuple, S.S#, P.P#).

- The problem of ensuring that the database does not include any invalid foreign key values is known as the referential integrity problem.

(foreign key relation-- > referencing relation, SP)

(primary key relation --> referenced relation, S, P, target relation)

- Referential diagram S <-- SP --> P, S <--S# SP P# --> P.

- A given foreign key and the corresponding primary key should be defined on the same underlying domain.

- No requirement that a foreign key be a component of the primary key of its containing relation.

DEPT(DEPT#, ..., BUDGET, ...)

EMP(EMP#, ..., DEPT#, ..., SALARY)

- EMP.DEPT# is a foreign key in relation EMP; however, it is not a component of the primary key EMP.EMP#.

- Relations R1 and R2 in the foreign key definition are not necessarily distinct.

  EMP (EMP#, ..., SALARY, ..., MANAGER#)

- Foreign-primary-key matches represent certain relationships between tuples.

∗The Referential Integrity Rule

"The database must not contain any unmatched foreign key values."

- If B references A, then A must exist.

- Problem: how exactly are such incorrect states to be avoid ?

  1) reject.

  2) accept the operation but to perform certain additional compensating operations (if necessary) in order to guarantee that overall result is still in a legal state. (cascade delete)

- Consider:
  1) S table: Insert (OK); Update/delete.
  2) SP table: Delete (OK); Update/insert.

- Three possibilities:
  1) Restricted (Ok or reject).
  2) Cascades (change other, too).
  3) Nullifies: the foreign key is set to null in all such matching shipments and the supplier is then deleted.

FOREIGN KEY (foreign key) REFERENCES target
(NULLS [NOT] ALLOWED)
(DELETE OF target effect)
(UPDATE OF target-primary-key effect)
where effect is RESTRICTED or CASCADES or NULLIFIES

- There is no explicit foreign key insert rule

Ex., an attempt to Insert (or update) a shipment (SP) tuple will succeed only if

(a) the supplier number in that tuple exists as a supplier number in relation S (or is null, if nulls are allowed).

(b) the part number ....

* Primary key and Foreign keys concepts and rules are applied to base relations.