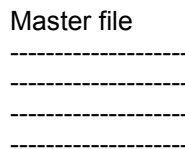
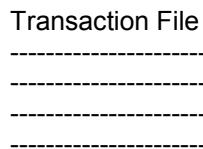
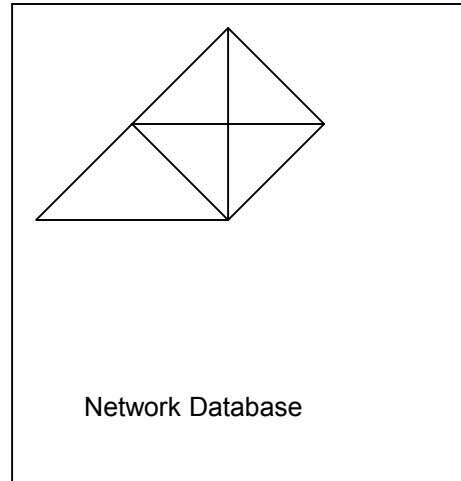
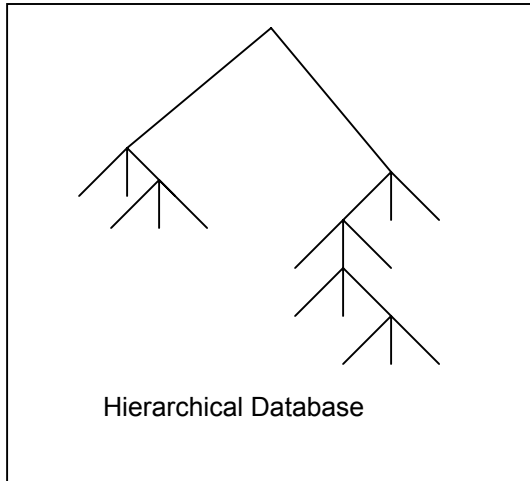
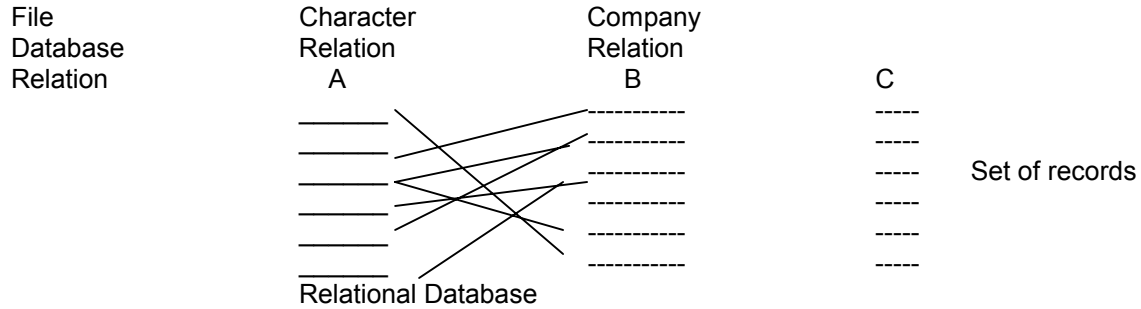


Introduction

DBMS 1957

A database can be defined as a set of Master files, organized & administered in a flexible way, so that the files in the database can be easily adapted to new unforeseen tasks!

Relation Vs. Relationship



There are 3 forms of database descriptions – the ANSI/SPARK, 1975 and so on...

1. Conceptual Schema (Conceptual View)
Is a machine-and-software independent description of the total database. It is also referred to as a **logical database**.
2. Internal Schema (Internal View)
Is a description of the **physical database!** It is close to the machine level & describes such things as file organization and access paths.
3. External Schema (User View)
Is a user oriented description of part of the database. Correspond to a way in which a program need to “view” the database.

- Since there are many purposes to which the data in a database maybe put, there will be many different external schema corresponding to different programs interpreting the database in particular ways.
- A DBMS is software used to read/write, maintain and provide among other things, security, and integrity for the data.

Facilities

- Standard methods used by DBMSs to implement relationships
- Data dictionary, docs, etc...
- Data independence (ie. the possibility of changing the physical database without having to make alternative to old programs that operate on the database.)

File

CODE	NAME	ADDRESS	TELEPHONE	SALARY
12345	XXXXXX	XXXXXX	XXXXXXXXX	YYYYYY

- Database languages
- Report generators/screen generators
- Recovery facilities
- Concurrency facilities
- File protection

Entities/Attributes

- An entity is an item or concept in the real world about which we want to report data. The data associated with an Entity is called **Attribute (field names in a record)**. An entity type is a classification of all the entities describing the same type of item or concept from the real world.
- A description of an Entity type in the conceptual database includes:
 - An unique name for each entity type.
 - A field containing a unique identifier for each individual
Entity → called a key
 - A description of all attributes or fields of the entity type
 - An indication of the number of occurrences of an entity type. The Database holds – known as the cardinal number of the entity type.

Define:

A conceptual schema maybe thought of as a model of the enterprise using it.

- Some attributes which maybe in an entity
 - Mandatory or optional depending on whether fields should or should not always have a value.
 - Single valued or multi-valued. Multi-valued-stored in repeating fields aggregate or simple.
 - Whether an attribute is formed from a combination of other attributes or NOT.

Relationships

- relationship is used “to describe” a connection between.

Relation: is used to designate a logical table describing a set of similar entities.

09/19/00

Project:

Flight Reservation/Scheduling

Agent	Airline Companies
Airport	Scheduling

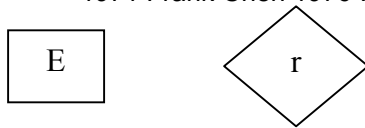
Conceptual Level.
File ≡ Relational ≡ Table

Data Model

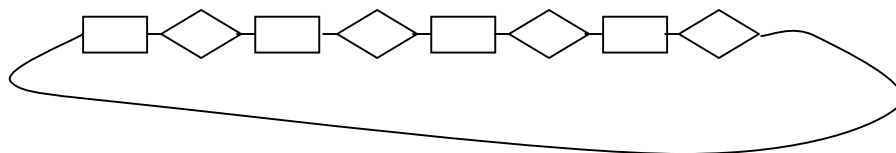
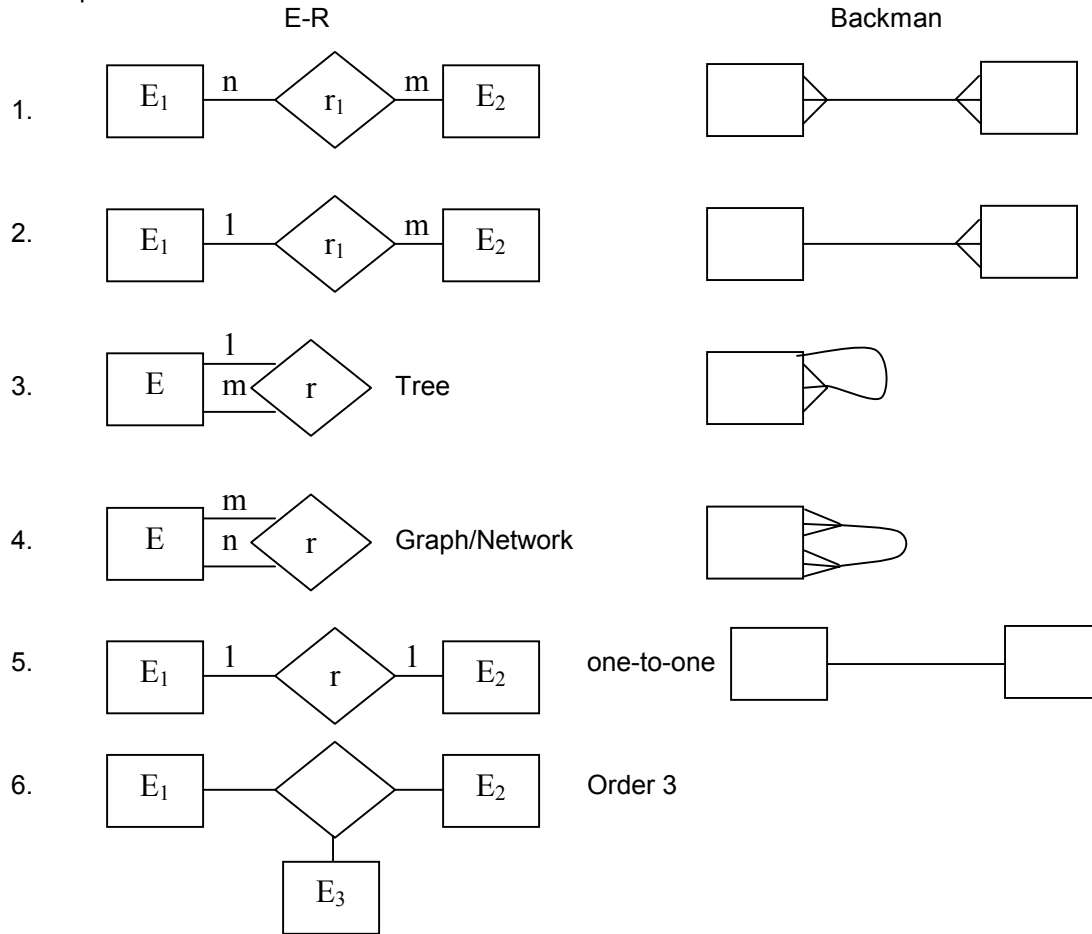
A data model is a method used to describe the Entity types & Relationships of Conceptual data.

- ❶ E-R Entity / Relationship
- ❷ Backman Diagrams
- ❸ Relational Model (see the paradigm)
- ❹ User View Diagram.

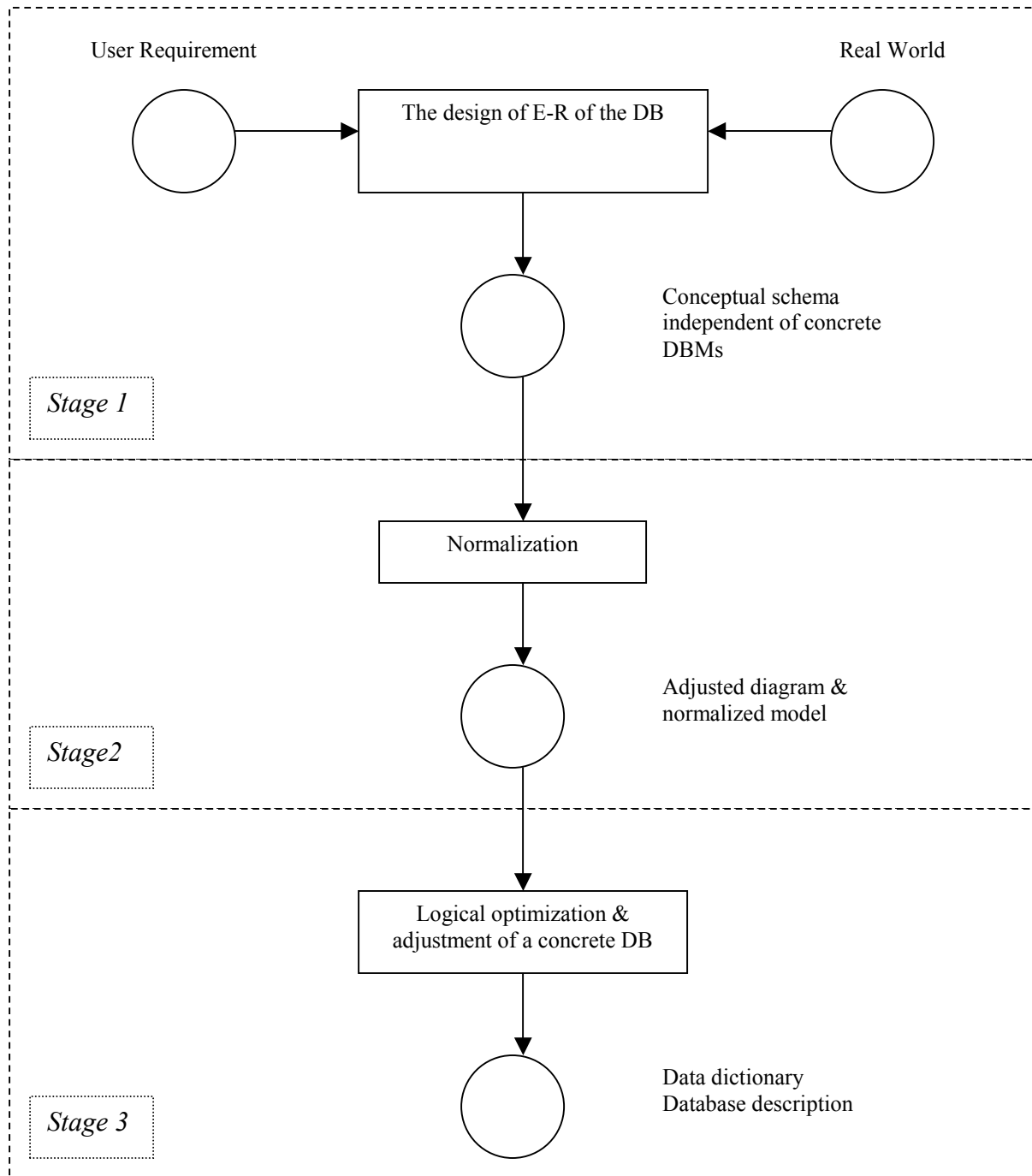
- ❶ The E-R is used for top-down analysis of views of a system.
 - 1971 Frank Chen 1976 ...



Example:



The design of the conceptual schema



Normalization

Normal form

- Formulate constraints on the structures of table, in the database to obtain a logical database that is like the “external world”.
- By applying different sets of constrains results in differently structured tables and normal forms.
- Terminology
 - Relational ≡ table ≡ file
 - Tuple → record
 - Attribute → a field
 - Domain → field’s value area

Example:

A sales person file (Before normalization)

Salesperson Number (SNR)	City	Sales Code (SC)	Product Number (PNR)	Amount (A)	PNR	A	PNR	A	PNR	A	PNR	A
S1	Athens	10	-	-	-	-	-	-	-	-	-	-
S2	Toronto	30	P1	200	P3	100	-	-	-	-	-	-
S4	Kingston	20	P5	200	P8	100	-	-	-	-	-	-
S5	Toronto	30	P1	50	P3	500	P4	800	P5	500	P8	1000

1NF (First Normal Form – FNF)

- Each tuple must contains an unique identifier.
- Tuples have only atomic values in their A’s (i.e. repeating groups are excluded)

The file after normalization → 1NF

SNR	City	SC	PNR	A
S1	Athens	10	-	-
S2	Toronto	30	P1	200
S2	Toronto	30	P2	100
S4	Kingston	20	P5	200
S4	Kingston	20	P8	100
S5	Toronto	30	P1	50
S5	Toronto	30	P3	500
S5	Toronto	30	P4	800
S5	Toronto	30	P5	500
S5	Toronto	30	P8	1000

Functional Dependency

- Assume the X & Y are fields in the same record (SC, City).
- Field Y is said to be functionally dependent on field X if and only if that for all pairs in the file that if they have the same value in field X, then they also have in Y.
- Field Y is said to be fully functional dependent on X if Y is functional dependent on X, and NOT functional dependent on any subset of X’s possible subfield.
- A field on which another field is fully functional dependent is called the determinant for that field.
- Field is said to be transitively functional dependent on X if there is some other field Z such that X determines the value of Z & Z determines the value of Y.

2NF (Second Normal Form – SNF)

- A relation is in 2NF if it is in 1NF and if all non-identifier fields are fully functional dependent on the records' identifier
- Example given with SC & City or with SNR & Name

SNF	Name
S1	CB
S2	BB
S4	RR
S5	PO

3NF (Third Normal Form)

- The relation is in 3NF if it is in 2NF and the record's non-identifier fields are NOT transitively dependent on the record's id field.
- The non-key fields must contain data that is attached to the id field, to the entire id and to nothing but the id field.
- For example, see 2NF.

3.5NF (Boyce/Codd Normal Form)

- Make use of determinant, has to be in 3NF.
- Any determinant **A** can be used as the tuples' id field
- So: The 3NF contains a constraint which the B/C does not.
- A relation is in the B/C normal form may well contain several different unique id fields, which the math deviation of the 3NF does not allow!
∴ the 3.5NF is more practical than the 3NF.

4NF

- Multi-valued dependency (between attribute)
- It holds between 2 **A**'s in a table if the second A can assume different value for given value in the first **A**.
- If a relation contains two multi-value dependencies, this may depend on or be independent of each other respectively.
- A relationship of order n (please see order 3 in the E-R diagram example).
- Will usually include many multi-valued dependence (MD).
- If these are independent of each other, the relationship of order n can be reduced to 1:n relationships and n:m relationships.

Example: Relationship of order 3

- A database contains a file for: vehicle dealers, vehicle manufacturers, vehicle types.

Dealer	Manufacturer
C.B.	Ford
C.B.	GM
O.O.	Ford
O.O.	GM

Dealer	Type
C.B.	Car
C.B.	Bus
O.O.	Car
O.O.	Truck

4NF

Multi-valued Dependency

- Between 2 attributes \Rightarrow if the second attribute can assume different values for a given value is the first attribute.
- MD either dependent on each other or independent on each other.
- Order n relationship.
- So we can say now that a relation (file) satisfies the 4th NF if it satisfies the 3NF(3.5NF) and the file contains second MD (dependent on each other).
- If a file does not satisfy the 4NF because it may contain independent MD \Rightarrow cannot split the file up!!
- Problems with maintenance and updating!

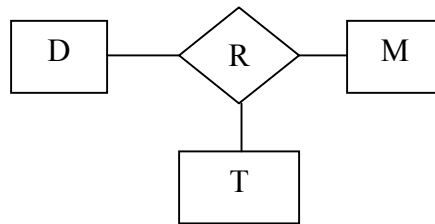
5NF

- Theoretical value.
- A relation R is in 5NF if and only if every join dependency (projection – join NF).
- Possible to reduce a R containing two MDs are dependent on each other

Example:

Assume that a db

Vehicle
Dealers
Manufacturers
Types



Note:

- N:M relation between dealer and manufacturers.
- N:M relation between dealer and types

a) Show that CB & MM each sell Ford & GM product

Dealer	Manufacturer
CB	Ford
CB	GM
MM	Ford
MM	GM

b) Show that CB sells cars & buses and MM sells cars & trucks.

Dealer	Type
CB	Car
CB	Bus
MM	Car
MM	Truck

- a) MD between dealer and manufacturer.
- b) MD between dealer and type.

- The 2 files can be implemented as a simple table with three columns.
 \Rightarrow the MDs are in the same table.

The 2 MDs are independent of each other if the “contents” of the first row mean that CB sell only Ford cars.

- if the MDs are independent, it is possible that CB doesn't sell Ford cars, only Ford buses.
- if the MD are dependent, CB sells Ford's cars.

4NF

Dealer	Manufacturer	Type
C.B.	Ford	Car
C.B.	Ford	Bus
C.B.	GM	Car
C.B.	GM	Bus
M.M.	Ford	Car
M.M.	Ford	Bus
M.M.	GM	Car
M.M.	GM	Bus

5NF

Dealer	Manufacturer
C.B.	Ford
C.B.	GM
M.M.	Ford

Dealer	Type
C.B.	Car
C.B.	Bus
M.M.	Truck

Manufacture	Type
Ford	Car
Ford	Bus
GM	Car
GM	Truck

Physical Database Structures

Args
Sequential
Random
Index
List

Keys	
Super Key:	An attribute (or combination of attribute) that uniquely identifies each entity
Candidate Key:	A minimal super key does not contain a subset of attributes that itself is a super key
Primary Key:	A candidate key selected to uniquely identify all other attribute values in any given row, cannot be null values (chosen types DB design)
Secondary Key:	An attribute (combination of attributes) used strictly for data retrieval purposes
Foreign Key:	An attribute (combination of attributes) in a table whose value must either match the primary key in another table or be null

Entity Sets

Weak entity set: An entity set does not have sufficient attributes to form a primary key.

Strong entity set: Has a primary key.

Integrity Rules

Entity integrity: No null value in primary key guarantee that each entity will have a unique identity

Referential Integrity: Foreign key should match another primary key or be null.

Relational Model

1. Selection
2. Projection
3. Union
4. Intersection
5. Join
6. Dimension
7. Difference

Consider 3 tables S, P & SP

S

SNR	SC	City
S1	10	Athens
S2	30	Toronto
S4	20	Kingston
S5	30	Toronto

P

PNR	PName	SIZE	Price
P1	Shirt	5	50
P3	Trousers	6	40
P4	Socks	6	3
P5	Blouse	8	20
P8	Blouse	5	25

SP

SNR	PNR	QTY
S2	P1	200
S2	P3	100
S4	P5	200
S4	P8	100
S5	P1	50
S5	P3	500
S5	P4	500
S5	P5	500
S5	P8	1000

Relational Language (RL)

Find temp = *algebraic expression* (finds in the file and then put it in a temp file)

Output *algebraic expression* (find it and output to screen)

(Combination of the 2 commands, Find something in file and output to screen and put into temp file)

Output temp = *algebraic expression*

1. Projection

Query: "Give all cities in S"

RL: find $T_1 = S(\text{City})$

Output: T_1

City
Athens
Toronto
Kingston

2. Selection (where)

Query: "What QTY of products P1 was sold by salesperson S5"

RL: find $T_2 = SP$ where SNR = "S5" and PNR = "P1"

Output: T_2

T2=

S5	P1	50
----	----	----

3. Union

Query: “which of the salesperson (SNE) are either resident of Toronto or have sold product P5”
RL: **find** T₃ = S(SNR) **where** City = Toronto
find T₄ = SP(SNR) **where** PNR = “P5”
Output: T₃ **Union** T₄

T ₃ :	<table border="1"><tr><td>S2</td></tr><tr><td>S5</td></tr></table>	S2	S5
S2			
S5			

T ₄ :	<table border="1"><tr><td>S4</td></tr><tr><td>S5</td></tr></table>	S4	S5
S4			
S5			

Output:	<table border="1"><tr><td>S2</td></tr><tr><td>S4</td></tr><tr><td>S5</td></tr></table>	S2	S4	S5
S2				
S4				
S5				

4. Intersection

Query: “which SNR are both resident of Toronto and have sold product P5”
RL: **find** T₃ = S(SNR) **where** city = “Toronto”
find T₄ = SP(SNR) **where** PNR = “P5”
Output: T₃ **intersect** T₄

Output:

S5

5. Difference (minus)

Query: “output SNR for the salesperson(s) who are resident in Toronto and who have not sold product P5”
 Note: The different of two tables is the set of entities that are member of the first of the tables but NOT in the other.
RL: **find** T₃ = S(SNR) **where** city = “Toronto”
find T₄ = SP(SNR) **where** PNR = “P5”
Output: **output** T₃ **minus** T₄

Output:

S2

6. Join

Note: Concatenation (?) of 2 rows which have a common field.

Query: Output a list of SNRs, resident of Toronto stating SNR, PNR & QTY.

RL: **find** T₆=S(SNR) **where** city='Toronto'
output join T₆ **with** SP **where** T₆(SNR)=SP(SNR)

Output

SNR
S2
S5

SNR	PNR	QTY
S2	P1	200
S2	P3	100
S5	P1	50
S5	P3	300
S5	P4	500
S5	P5	500
S5	P8	100

7. Product

Note: Cartesian product of 2 tables

Query: Produce a file containing all combinational possibilities of SNRs and PNRs

RL: find $T_7 = \text{MULT } S(\text{SNR}) \text{ with } P(\text{PNR})$

Output:

SNR	PNR
S1	P1
S1	P3
S1	P4
S1	P5
S1	P8
S2	P1
S2	P3
S2	P4
S2	P5
S2	P8
S4	P1
S4	P3
S4	P4
S4	P5
S4	P8
S5	P1
S5	P3
S5	P4
S5	P5
S5	P8

8. Division

Query: “Find the SNRs who have sold all products”

Note: 1) Table SP(SNR, PNR) can be divided by table P(PNR) as column PNR is shared by both tables.

2) Divider ÷ divisor = quotient

RL: find quotient = **divide** SP(SNR, PNR) **by** (PNR)

Output:

Divider		Divisor	Remainder		Quotient
SP(SNR, PNR)		PNR	SNR	PNR	SNR
SNR	PNR	P1	S2	P1	S5
S2	P1	P3	S2	P3	
S2	P3	P4	S4	P5	
S4	P5	P5	S4	P8	
S4	P8	P8			
S5	P1				
S5	P3				
S5	P4				
S5	P5				
S5	P8				

The R.L is a complete Relational Language.

DDL Data Definition/Description Language

Create Table, Alter Table, Drop Table
 Create View, Drop View
 Create Index, Drop Index

Create table *table-name*
 (column definition [, column definition, ...]
 [primary key definition]....
 [alternate key definition] ...
 [foreign key definition] ...
 [other parameters]);

Examples:

Create table S
 (SNR char(5) NOT NULL,
 sname char(20),
 s-c smallint,
 city char(5),
 primary key (SNR));

Create table P
 (PNR char (5) NOT NULL,
 pname char(20),
 size smallint,
 price integer,
 primary key (PNR));

Create table SP
 (SNR char(5) NOT NULL,
 PNR char(5) NOT NULL,
 Qty integer,
 primary key (SNR),
 primary key (PNR),
 foreign key SFK(SNR) references S,
 foreign key PFK(PNR) references P);

Copy of a table:

Create table *tablename* **LIKE** *tablename*

Eg.

create table SCOPTY **LIKE** S

Alter Table

Alter table *tablename*
ADD *column-name datatype* [NOT NULL], etc...

Example:

Alter table S **ADD** address char(20);

DROP table

Drop table *tablename*;

Eg.

Drop table S;

DML Data Manipulation Language

SELECT, INSERT, DELETE

Example:

1. Query: output sorted list of cities

SQL:

(Selects all city no duplicates from table S, in ascending order)
SELECT DISTINCT city FROM S ORDER BY city;

Athens
Kingston
Toronto

(Selects all city no duplicates from table S, in descending order)
SELECT DISTINCT city FROM S ORDER BY city DESC;

Toronto
Kingston
Athens

(Selects all city from table S)
SELECT city FROM S;

Athens
Toronto
Kingston
Toronto

2. Query: display a list of SNR salesperson reside in Toronto

SQL: (* - all fields/columns)

SELECT * from S where city='Toronto';

SNR	sname	SC	city
S2	CB	30	Toronto
S5	BB	30	Toronto

SELECT * from S where city='Toronto' ORDER BY sname DESC;

3. Query: Which SNR(s) are either resident of Toronto or have sold product P5.

SQL:

SELECT SNR FROM S WHERE city='Toronto'
UNION
SELECT SNR FROM SP WHERE PNR='P5';

SNR
S2
S4
S5

(note there are more than one way to do this)

4. Query: Display a list of sname, SNRs, PNRs and Qty sold.

SQL:

SELECT sname, S.SNR, PNR, QTY **FROM** S, SP **WHERE** S.SNR=SP.SNR;

Output:

sname	SNR	PNR	QTY
CB	S2	P1	200
CB	S2	P3	100
BB	S4	P5	200
BB	S4	P8	100
OO	S5	P1	50
OO	S5	P3	300
OO	S5	P4	500
OO	S5	P5	500
OO	S5	P8	100

5. Complex (from Relational Calculus)

Query: display a list of PNRs that where sold in a size that is 3 size larger and which has the same pname.

SQL:

(there is a space between P x)

SELECT x.PNR **FROM** P x, P y **where** x.pname = y.pname **AND** x.size=ysize+3;

(another way to write this, but may not work for all Databases)

SELECT x.PNR **FROM** P **AS** x, P **AS** y **where** x.pname = y.pname **AND** x.size=ysize+3;

X.PNR
P5

6. Query: display a list of all pairs of SNRs with different scodes.

SQL:

SELECT s.SNR, y.SNR **FROM** S x, S y **WHERE** (x.SC < y.SC);

Output:

X.SNR	y.SNR
S1	S2
S1	S4
S1	S5
S4	S2
S4	S5

7. Query: Display PNR of “blouses” with price range 20 to 60.

SQL:

SELECT PNR **FROM** P **WHERE** pname=”blouse” **AND** price **BETWEEN** 20 **AND** 60;

(Alternatively)

SELECT PNR **FROM** P **WHERE** pname=”blouse” **AND** (price >=20 **AND** price <= 60);

PNR
P5
P8

8. Query: Display all (everything: *) with SC (5, 8,10 or 15) and city not equal Toronto & Athens.

SQL:

```
SELECT * FROM S WHERE SC IN (5, 8, 10, 15) AND city NOT IN ('Toronto', 'Athens');
```

(Alternatively)

```
SELECT * FROM S WHERE (SC='5' OR SC='8' OR SC='10' OR SC='15') AND (city <> 'Toronto' AND city <> 'Athens');
```

Output: NO RECORDS

LS's important note:

As I know, there is no operator EXIST. But there is an operator EXISTS in SQL
 So if you try creating a query or view from Access, SQL Server, Oracle, etc... EXIST may not work.
 If so, change the below #9 and #10 to EXISTS and the view/query should work.

9. Exist Operator

Note: The logical expression with the operator exist has value TRUE if the arguments of exist is not an empty table and the where is a complex expression.

Query: Display PNR(s) & pname(s) (PNR, pname) of products sold.

SQL:

```
SELECT PNR, pname FROM P WHERE EXIST SELECT * FROM SP WHERE SP.PNR=P.PNR;
```

Output:

PNR	Pname
P1	Shirt
P3	Trousers
P4	Socks
P5	Blouse
P8	Blouse

10. Query: Display the set of SNRs who have sold same or all products

SQL:

```
SELECT * FROM S WHERE NOT EXIST (
    SELECT * FROM P WHERE NOT EXIST (
        SELECT * FROM SP WHERE SP.PNR=P.PNR AND SP.SNR=S.SNR));
```

Output:

SNR	SC	City
S5	30	Toronto