PHYSICAL DESIGN
relational structures

AGGREGATION: ADDING KEYS.

EXAMPLE:

***type* motor vehicle =**
manufacturer, model, year, price, fuel,

a relation is generated by adding key attribute vehicle id:

***relation* motor vehicle**
(vehicle id, manufacturer, model, year, price, fuel).

If the composite type contains references to other types, foreign keys must be included for these types as well:

***type* dispatch = motor vehicle, destination, cargo.**

The equivalent relation is:

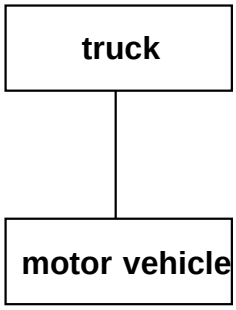
***relation* (dispatch id, vehicle id, destination, cargo).**

PHYSICAL DESIGN
relational structures

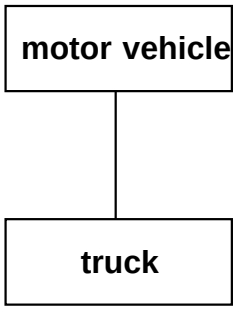
GENERALIZATION: THREE ALTERNATIVES.

type **motor vehicle = manufacturer, model, year, ...**
type **truck = [motor vehicle], loading_capacity.**

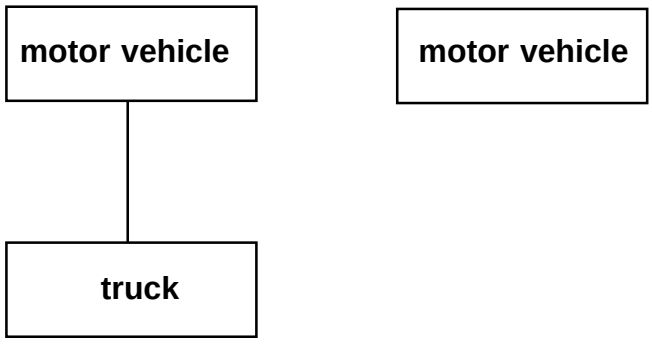
1.



2.

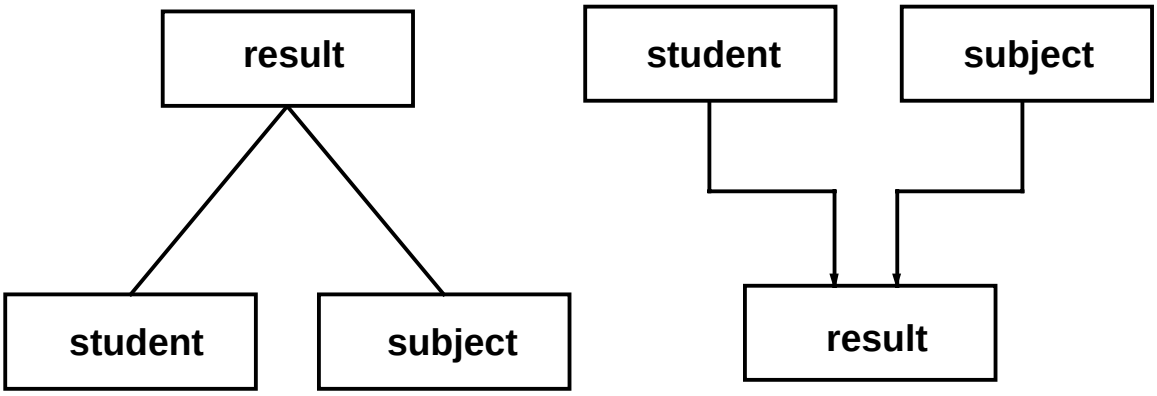


3.

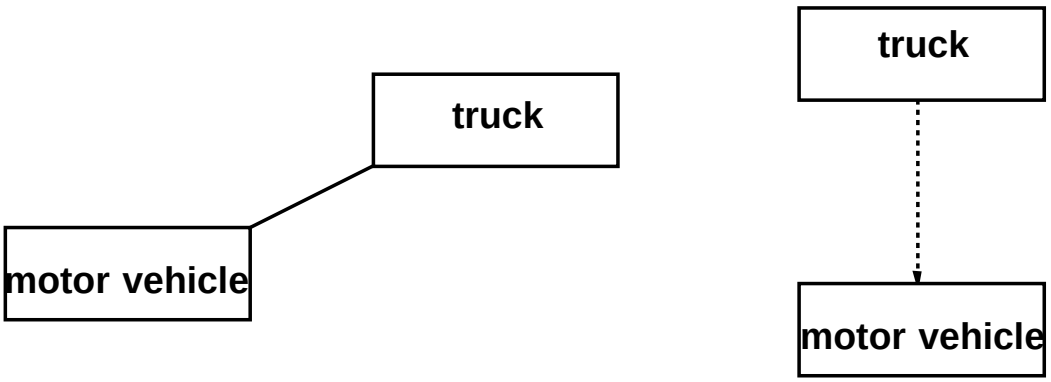


PHYSICAL DESIGN
network structures

AGGREGATION: set membership.



SPECIALIZATION: sometime membership.



PHYSICAL DESIGN
semantic versus relational

SEMANTIC: including relatability.

base **name (A20).**
base **town (A20).**
base **faculty (A20).**
base **description (A40).**
base **mark (A1).**
type **student (I6) = name, town, faculty**
type **subject (I4) = description, lecturer_name**
type **result (I8) = student, subject, mark.**

RELATIONAL: without referential constraints.

```
CREATE TABLE STUDENT (
      S# (CHAR(6), NONULL),
      NAME (CHAR(20)),
      TOWN (CHAR(20))
      FACULTY (CHAR(20)) )
CREATE TABLE SUBJECT (
      U# (CHAR(4), NONULL),
      DESCRIPTION (CHAR(40)),
      DNAME (CHAR(20)) )
CREATE TABLE RESULT (
      R# (CHAR(6), NONULL),
      S# (CHAR(6), NONULL),
      U# (CHAR(4), NONULL),
      MARK (CHAR(1)) ).
CREATE UNIQUE INDEX XS ON STUDENT (S#)
CREATE UNIQUE INDEX XU ON SUBJECT (U#)
CREATE UNIQUE INDEX XR ON RESULT (R#).
```

PHYSICAL DESIGN
manipulation

Example S1: Select students from Edinburgh.

XPLAIN:

```
get student where town = "Edinburgh".
```

SQL:

```
SELECT *
FROM STUDENT
WHERE TOWN = "Edinburgh".
```

Example S2: Select results in the subject Physics.

XPLAIN:

```
get result where subject its description =
"Physics".
```

SQL:

```
SELECT *
FROM RESULT
WHERE U# IN
    (SELECT U#
    FROM SUBJECT
    WHERE DESCRIPTION = "Physics").
```

OR:

```
SELECT R.R#, R.S#, R.U#, R.MARK
FROM RESULT R, SUBJECT S
WHERE R.U# = S.U#
AND S.DESCRIPTION = "Physics".
```

PHYSICAL DESIGN
manipulation (continued)

Example S3: Select students who did not pass the subject Physics.

XPLAIN:

```
extend student with fail =
        nil result
        where mark ≠ "F" and
                subject its description = "Physics"
        per student.
get student where fail.
```

SQL:

```
CREATE VIEW FAIL (S#) AS
        SELECT S#
        FROM STUDENT
        WHERE S# NOT IN
                (SELECT S#
                FROM RESULT
                WHERE MARK ≠ "F"
                        AND U# IN
                        (SELECT U#
                        FROM SUBJECT
                        WHERE DESCRIPTION = "Physics")).
SELECT *
FROM STUDENT
WHERE S# IN
        (SELECT S#
        FROM FAIL).
```

Example S4: Select students with more than eight passes.

XPLAIN:

```
extend student with number =
        count result
        where mark ≠ "F"
        per student.
get student where number > 8.
```

SQL:

```
CREATE VIEW CANDIDATE (S#, NUMBER) AS
        SELECT S.S#, COUNT (*)
        FROM STUDENT S, RESULT R
        WHERE S.S# = R.S# AND R.MARK ≠ "F"
        GROUP BY S.S#.
SELECT *
FROM STUDENT
WHERE S# IN
        (SELECT S#
        FROM CANDIDATE
        WHERE NUMBER > 8).
```

HOWEVER !!!

Example S5: Select students with less than three passes.

PHYSICAL DESIGN
manipulation (continued)

Example S5: Select students with less than three passes.

XPLAIN:

```
extend student with number =
        count result
        where mark ≠ "F"
        per student.
get student where number < 3.      (cf. 5: > 8 ⇒ < 3)
```

SQL:

```
CREATE VIEW CANDIDATE (S#, NUMBER) AS
        SELECT S.S#, COUNT (*)
        FROM STUDENT S, RESULT R
        WHERE S.S# = R.S# AND R.MARK ≠ "F"
        GROUP BY S.S#.
SELECT *
FROM STUDENT
WHERE S# IN
        (SELECT S#
        FROM CANDIDATE
        WHERE NUMBER < 3)
UNION
SELECT *
FROM STUDENT
WHERE S# NOT IN
        (SELECT S#
        FROM RESULT).
```

BE AWARE OF EMPTY SETS !!!!!

PHYSICAL DESIGN
manipulation (continued)

Example S6: Select students with the maximum number of passes.

XPLAIN:

```
extend student with number =
        count result
        where mark ≠ "F"
        per student.
value maximum =
        max student its number.
get student where number = maximum.
```

SQL:

```
CREATE VIEW CANDIDATE (S#, NUMBER) AS
        SELECT S.S#, COUNT (*)
        FROM STUDENT S, RESULT R
        WHERE S.S# = R.S# AND R.MARK ≠ "F"
        GROUP BY S.S#.
CREATE VIEW MAXIMUM (VALUE) AS
        SELECT MAX (NUMBER)
        FROM CANDIDATE.
SELECT *
FROM STUDENT
WHERE S# IN
        (SELECT S#
        FROM CANDIDATE, MAXIMUM
        WHERE NUMBER = VALUE).
```

In SQL is assumed that at least one student has a result !

PHYSICAL DESIGN
exercises

EXERCISE 2

Consider the following relational model for a data dictionary:

***relation* table**
(table name, table data)
***relation* column**
(table name, column name, column type, length).

Why can the requirement 'each table contains at least one column' never be an invariant for this data dictionary?

EXERCISE 3

Assume following structure for a semantic model data dictionary:

***type* type = kind, representation**
***type* attribute = composite_type, type**
***type* special type = [type], prefix.**

Provide the corresponding abstraction hierarchy and name a few problems the above structure could cause.

PHYSICAL DESIGN
exercises (continued)

EXERCISE 4

Consider a conceptual model with only one type of abstraction. Which abstractions can consequently be found in the data dictionary model? Assume that the data dictionary model is regarded as a database once more. What is the structure of this conceptual model? What can we conclude therefore?

EXERCISE 5

Why is following data dictionary structure unsuitable?

***relation* table (table name, table data, key)**
***relation* attribute (table name, column name)**
***relation* column (column name, column type, length).**

PHYSICAL DESIGN
exercises (continued)

EXERCISE 6

Convert the definitions of the semantic model below to SQL definitions.

type **patient**	**= name, address, town**
type **physician**	**= name, extension, department**
type **department**	**= internal_address, extension**
type **treatment type**	**= description, hourly_rate**
type **treatment**	**= patient, physician, treatment type, date, minutes_duration**
type **admission**	**= patient, physician, admission_date, release_date.**

EXERCISE 7

Formulate the following queries by using SQL.

a **Select patients treated by a physician in department D9.**

b **Select patients admitted to hospital by a physician in department D9.**

c **Select patients having undergone treatment, but who were never admitted to hospital.**

d **Select patients having undergone treatment by a physician, who also treated a patient treated by a physician in department D9.**

e **Determine total treatment costs per patient per admission.**

PHYSICAL DESIGN
SQL exercises (continued)

```
CREATE TABLE PATIENT (
        PNR (NUMBER (5), NONULL),
        NAME (CHAR (20)),
        ADDRESS (CHAR (20)),
        TOWN (CHAR(20) )
CREATE TABLE PHYSICIAN (
        PHNR (NUMBER (5), NONULL),
        NAME (CHAR (20)),
        EXTENSION (NUMBER (5)),
        DNR (CHAR (2)) )
CREATE TABLE DEPARTMENT (
        DNR (CHAR (2), NONULL),
        ADDRESS (CHAR (20)),
        EXTENSION (NUMBER (5)) )
CREATE TABLE TREATMENTTYPE (
        TTNR (NUMBER (5), NONULL),
        DESCRIPTION (CHAR (20)),
        RATE (NUMBER (4)) )
CREATE TABLE TREATMENT (
        TNR (NUMBER (5), NONULL),
        PNR (NUMBER (5)),
        PHNR (NUMBER (5)),
        TTNR (NUMBER (5)),
        DATE (CHAR (6)),
        DURATION (NUMBER (4)) )
CREATE TABLE ADMISSION (
        ANR (NUMBER (8), NONULL),
        PNR (NUMBER (5)),
        PHNR (NUMBER (5)),
        STARTDATE (CHAR(6)),
        ENDDATE (CHAR(6)) )
CREATE UNIQUE INDEX XP ON PATIENT (PNR)
CREATE UNIQUE INDEX XPH ON PHYSICIAN (PHNR)
CREATE UNIQUE INDEX XD ON DEPARTMENT (DNR)
CREATE UNIQUE INDEX XTT ON TREATMENTTYPE (TTNR)
CREATE UNIQUE INDEX XT ON TREATMENT (TNR)
CREATE UNIQUE INDEX XA ON ADMISSION (ANR)
```

PHYSICAL DESIGN
SQL exercises (continued)

EXERCISE 7 d: Nested expression

```
SELECT P1.PNR, P1.NAME, P1.ADDRESS, P1.TOWN
FROM PATIENT P1
WHERE EXISTS (
      SELECT *
      FROM PHYSICIAN PH1
      WHERE EXISTS (
            SELECT *
            FROM TREATMENT T1
            WHERE PH1.PHNR = T1.PHNR
            AND P1.PNR = T1.PNR)
            AND EXISTS (
                  SELECT *
                  FROM TREATMENT T2
                  WHERE PH1.PHNR = T2.PHNR
                  AND EXISTS (
                        SELECT *
                        FROM PATIENT P2
                        WHERE P2.PNR = T2.PNR
                        AND EXISTS (
                              SELECT *
                              FROM PHYSICIAN PH2
                              WHERE PH2.DNR = "D9"
                              AND EXISTS (
                                    SELECT *
                                    FROM TREATMENT T3
                                    WHERE T3.PHNR =
                                    P2.PHNR
                                    AND P2.PNR = T3.PNR)))))
```

PHYSICAL DESIGN
SQL exercises (continued)

EXERCISE 7 d: Flat join expression

```
SELECT P1.PNR, P1.NAME, P1.ADDRESS, P1.TOWN
FROM PATIENT P1, PHYSICIAN PH1, TREATMENT T1,
     PATIENT P2, PHYSICIAN PH2, TREATMENT T2,
     TREATMENT T3
     WHERE T1.PHNR = PH1.PHNR
     AND T1.PNR = P1.PNR
     AND T2.PHNR = PH1.PHNR
     AND T2.PNR = P2.PNR
     AND PH2.DNR = "D9"
     AND T3.PHNR = PH2.PHNR
     AND T3.PNR = P2.PNR
```

PHYSICAL DESIGN
SQL exercises (continued)

EXERCISE 7D:
Derived from semantic solution (cf. exercise 11.7.7)

```
CREATE VIEW I (PNR) AS
    SELECT PNR FROM PATIENT
    WHERE PNR IN
        (SELECT PNR FROM TREATMENT
        WHERE PHNR IN
            (SELECT PHNR FROM PHYSICIAN
            WHERE DNR = "D9"))

CREATE VIEW II (PHNR) AS
    SELECT PHNR FROM PHYSICIAN
    WHERE PHNR IN
        (SELECT PHNR FROM TREATMENT
        WHERE PNR IN
            (SELECT PNR FROM I))

CREATE VIEW III (PNR)
    SELECT PNR FROM PATIENT
    WHERE PNR IN
        (SELECT PNR FROM TREATMENT
        WHERE PHNR IN
            (SELECT PHNR FROM II))

SELECT PNR, NAME, ADDRESS, TOWN
FROM PATIENT
WHERE PNR IN
    (SELECT PNR FROM III)
```