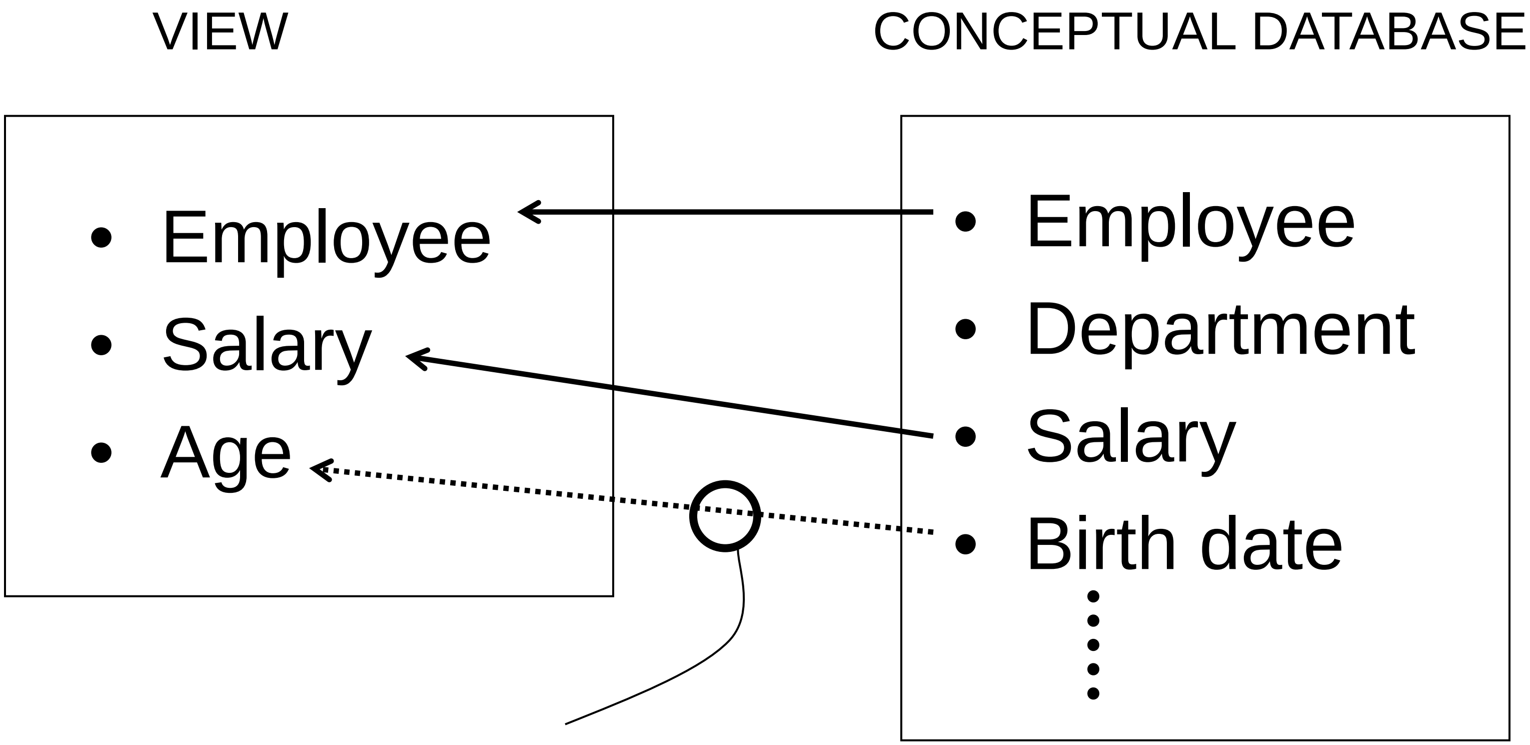
VIEWS

- A **virtual relation** that is defined from other pre-existing relations 🡐Called the "defining relations" of the view
- A view supports multiple user perspectives on the database corresponding to different information organizations, avoiding the need for **data duplication** or information **consistency problems**
- Additional motivation: security (privacy concerns, users need only access/modify selected attributes in the data without being able to access the other attributes)

Differences between a **Conceptual Database** and a **View**



VIEW

- Employee
- Salary
- Age

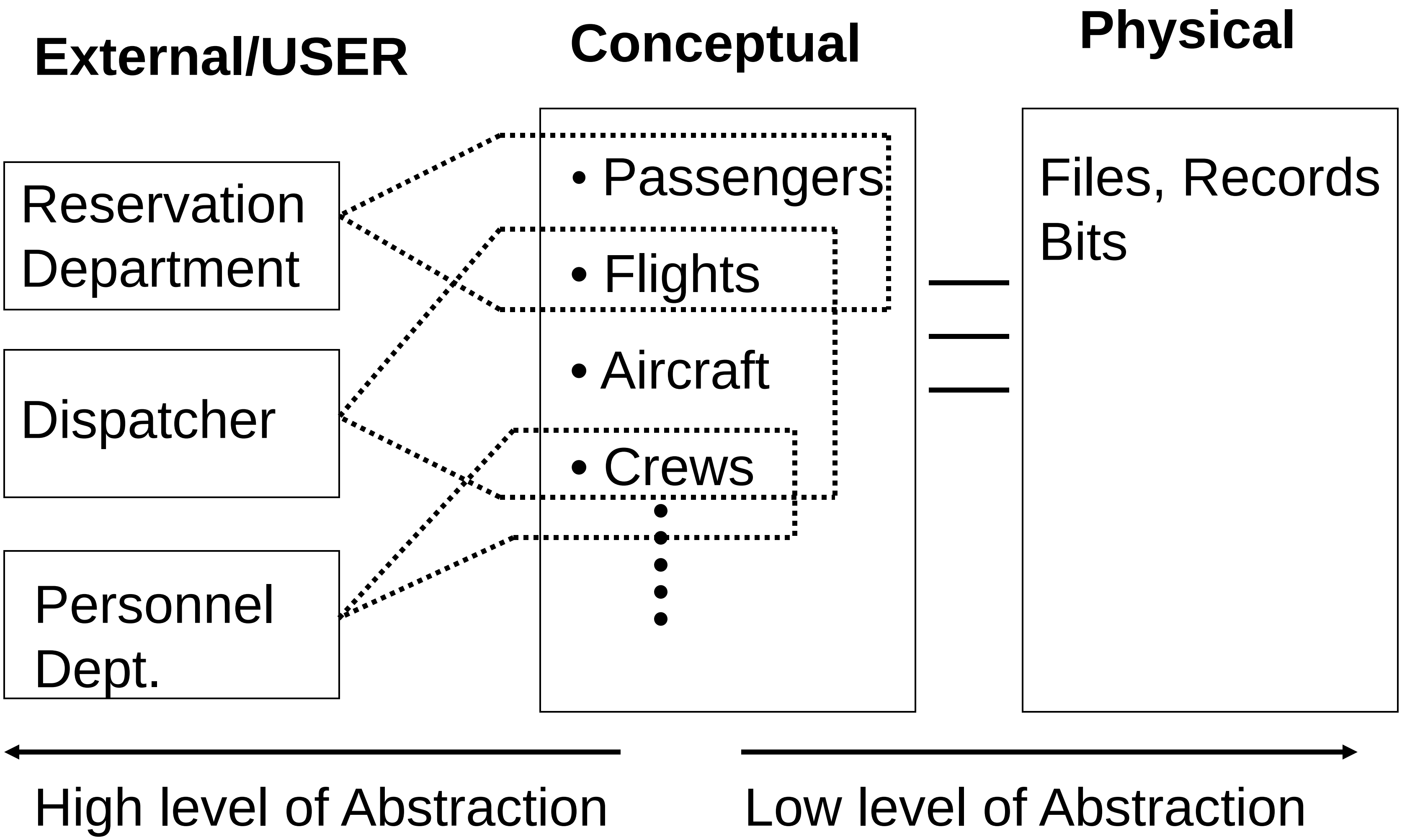
CONCEPTUAL DATABASE

- Employee
- Department
- Salary
- Birth date

⋮

Constructable, but not actually present in the database

EXAMPLE: Airline Databases



External/USER

Conceptual

Physical

Reservation Department

Dispatcher

Personnel Dept.

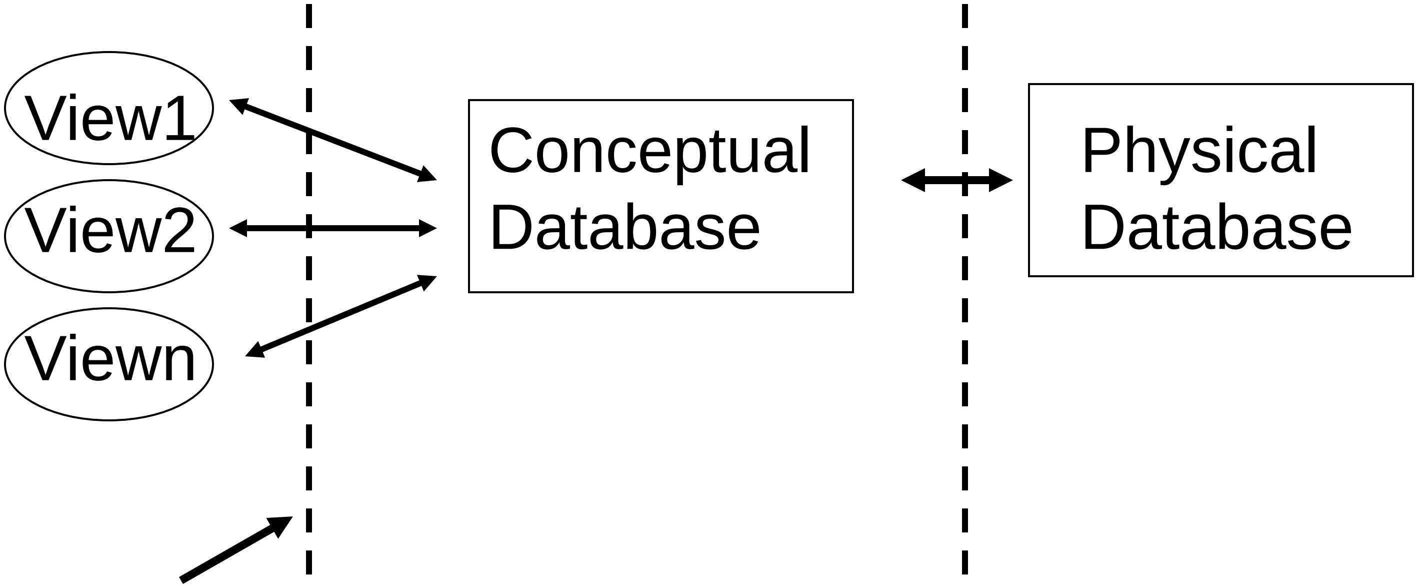
- Passengers
- Flights
- Aircraft
- Crews

Files, Records Bits

High level of Abstraction

Low level of Abstraction

DATA INDEPENDENCE



Logical Data Independence

- Many modifications of conceptual scheme can be made without affecting views
- No Changes in application programs necessary

Physical Data Independence

- Physical schema can be changed without alerting the conceptual level
- Allows for tuning

Analogy from the programming language world

View level

Function f(i)

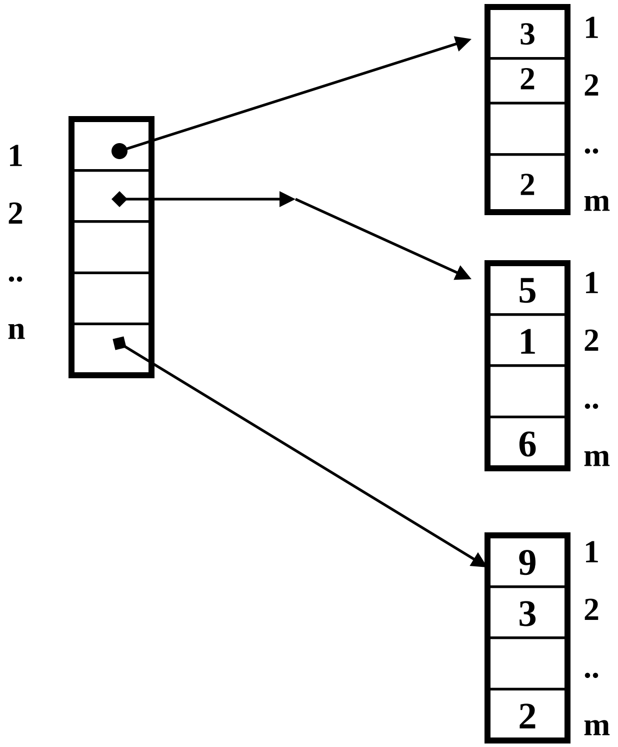
$f(i)=\sum^{m}_{j=1}A(i,j)$

Concept Level

Integer Array

A[1..n,1..m]

Physical Level



VIEW DEFINITIONS

Views are defined by a query that generates the desired virtual relation from existing relations:

In Relational Algebra:

	From Department		From Employee	
DEPHEADS	**DNAME**	**DNO**	**FNAME**	**LNAME**
	Pebody	5	Robin	Wang
	Admin	4	Jenifer	Veallau
	CS	1	James	Borg

Create view DEPHEADS **As**

$$\pi_{\text{DNAME,DNUMBERS,FNAME,LNAME}}$$

$$(\text{EMPLOYEE} \bowtie_{\text{MGRSSN=SSN}} \text{DEPARTMENT})$$

→ ***Views typically include selected projections with optional selects and joins***

VIEWS IN SQL

```
CREATE VIEW      DEPHEADS   AS
         SELECT    DNAME,DNUMBER,FNAME,LNAME
         FROM      DEPARTMENT,EMPLOYEE
         WHERE     DEPARTMENT.MGRSSN=EMPLOYEE.SSN
```

Defining SQL Query

Attribute in view relation

Defining relation

The SQL statement defining view are typically executed at query time, thus additions/changes in the defining/base relations are reflected in the virtual relation (view) transparently.

Allow Query a views:

```
SELECT   FNAME,LNAME
FROM     DEPHEADS
WHERE    DNUMBER=5
```

The information is a virtual relation is always “up to date” (automatically reflect database update)

Complex Views

- Views may include complex calculations

Remaining attributes

```
CREATE VIEW     EMP_AGE(LNAME,AGE) AS
        SELECT LNAME,MONTHS-BETWEEN(SYSDATE,BDATE)/12
        FROM    EMPLOYEE
```

Built op.

LNAME	AGE
SMITH	40.86
WANG	50.91
ZELAYA	38.12

See Chapter 7

- Views creation/definition may contain aggregate operation

```
CREATE VIEW  DEPT-INFO(DEPT_NAME,NUM_EMPS,TOTAL_SAL) AS
SELECT  DNAME,COUNT(*),SUM(SALARY)
FROM     DEPARTMENT, EMPLOYEE
WHERE   DNUMBER=DNO
GROUP BY DNAME;
```

DEPT_NAME	NUM EMPLOYER	TOTAL SALARY
RESEACHER	4	135,000
ADMIN	3	93,000
HEADQUATER	1	35,000

Changes to the database via views

```
UPDATE DEPHEAD
SET     DNAME = ‘research’
WHERE   LNAME = ‘Wallace’ OR LNAME = ‘SMITH’
```

Rename all departments manager by ‘Wallace’ or ‘Smith’ to ‘Research’

Syntax:

```
UPDATE   <VIEW-NAME>
SET      <LIST OF CHAGES TO VIEW ATTRIBUTES>
WHERE    <condition based on view attributes>
```

mapped to necessary updates in the defining relation

OK if simple name change

PROBLEMS WITH VIEWS

Insert into a view based on **a join**

Insert into DEPHEADS
Values(‘SALES”,6,’John’,’Wilson’)

DEPARTMENT

DNAME	DNO	MGRSSN	LOC
RESEARCH	5	33344455	401
ADMIN	4	98316738	201
SALES	6	……	

……

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRES	SAL
John	Smith	419324	11,July	223333	30000
James	Bary	123123	10 Nov	111222	55000
John	Wilson	------	-----	----	----

Null values in the fields projected out of the defining relations by the view.

ANOMOLY

```
SELECT LNAME
FROM    DEPHEADS
WHERE DNAME=‘SALES’
```

→

```
SELECT EMPLOYEE.LNAME
FROM DEPARTMENT,EMPLOYEE
WHERE MGRSSN=SSN
         AND DNAME=‘SALES’
```

Fields null result. Join fail because null join attribute

Additional Problem with views

```
UPDATE  DEPT_INFO

SET        TOTAL_SAL = 100000

WHERE    DNAME = 'RESEARCH'
```

Problem when view attribute is defined as an aggregate quantity

→ how can the constraint

sum(salary)=100000

be realized as an update on the individual salary attributes for dept with > 1 employee

RESTRICTIONS ON VIEWS

to avoid consistency problems

- In general, **updates** are only allows when there is only **one possible** **update** in the base relation to accomplish the view update.

1). A view with a **single defining table** is updatable, if

a) The view attribute contain the **primary key** and all other "not null" attributes.

(still problem of nulls in the defining relations)

2). Views defines using

--- joins

--- grouping

---aggregate functions

generally not updatable

⇒ But generally no restrictions on read-only views

View Implementation Issue

Strategy #1: **QUERY MAPPING** Oracle approach

Convert query on view to query on base relation

Problem: may be inefficient if the view involves complex calculation like aggregate function.

Strategy #2: **VIEW MATERIALIZATION**

Create temporary table to reflect the view structure

--- efficient if many queries to few updates

Temporary table must be updated (recomputed) if updates to the defining relations

--- full recomputation costly

--- minimal update difficult to determine

--- goal of avoiding data duplication

View Implementation Issue

(Runtime macro)

Strategy #1: QUERY MAPPING

Oracle approach

Convert query on view to query on base relation

Problem: may be inefficient if the view involves complex calculation like aggregate function.

Strategy #2: VIEW MATERIALIZATION

Cache

Create temporary table to reflect the view structure

--- efficient if many queries to few updates

Temporary table must be updated (recomputed) if updates to the defining relations

--- full recomputation costly

--- minimal update difficult to determine

--- fail to avoid relational count of eliminating data duplication