Ch 3. Relational Model

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

SKKU VLDB Lab.

(http://vldb.skku.ac.kr/)

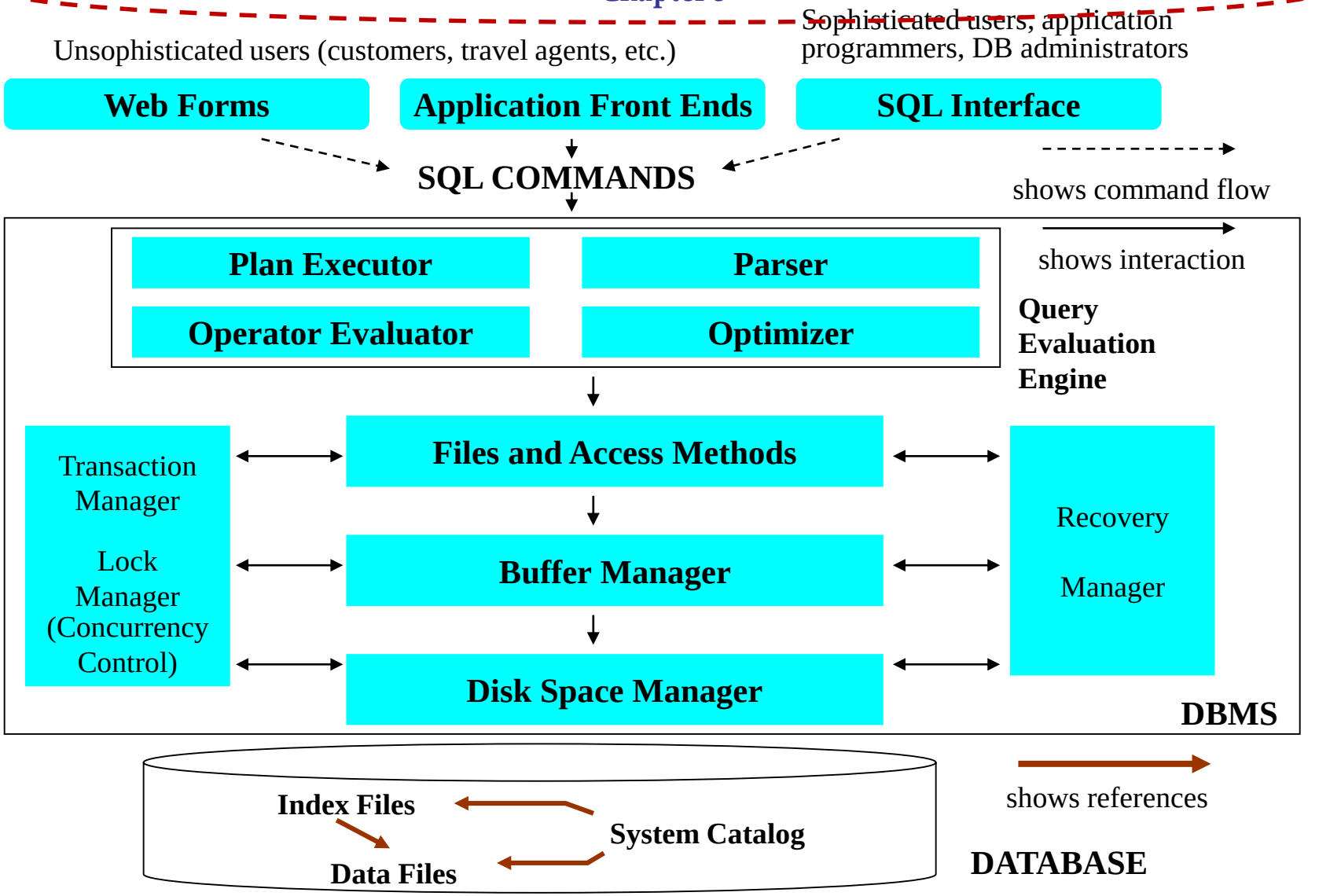
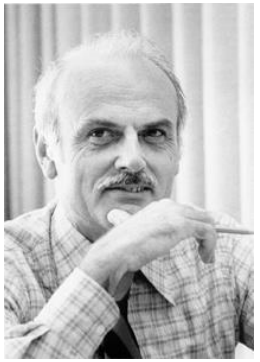


Figure 1.3 Anatomy of an RDBMS

What is (Relational) Data Model?

- Structure
  - ***Structural Part:*** DDL (Data Definition Language):
- Operations
  - ***Manipulative Part:*** DML (Data Manipulation Language):
- Constraints
  - ***Integrity Part:*** DCL (Data Control Language):



Information Retrieval — P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

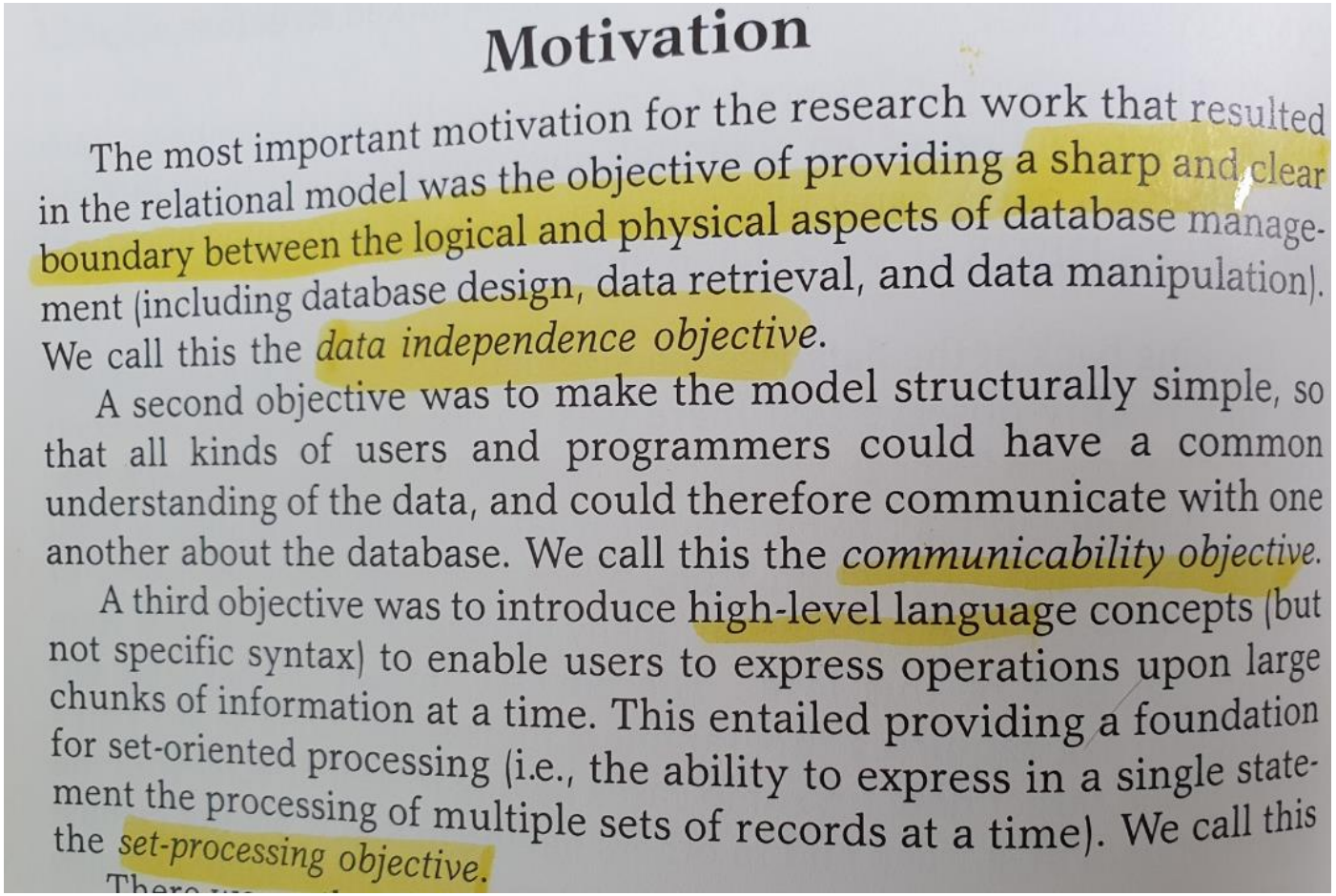
Relational Model of Data Large Shared Data Banks, June 1970, CACM

Very Large Data Bases

Why Study the Relational Model?

- Most widely used model. WHY?
  - "The relational model .. simple data representation and elegant query language …. even novice users to understand the contents of a database and to express complex queries." (p 58 in textbook)
  - Commercials: IBM, Microsoft, Oracle, SAP Hana, Sybase, Altibase, Tibero etc.
  - Open sources: MySQL/MariaDB, Postgres, SQLite, Berkeley DBMS
- "Legacy systems" in older models (e.g. network, hierarchical)
  - e.g., IBM's IMS
- Competitors: Object-Oriented model , XML, **NoSQLs (**JSON documents)
  - Pure OO-DBMS: ObjectStore, Versant, Ontos
  - A hybrid (or synthesis) data model: Object-Relational model
    - ✓ Informix Universal Server, UniSQL, O2, Oracle, DB2
    - ✓ Despite its several pros, OO is quite complex for undergraduates to learn and to use

Relational Data Model: Motivation



Motivation

The most important motivation for the research work that resulted in the relational model was the objective of providing a sharp and clear boundary between the logical and physical aspects of database management (including database design, data retrieval, and data manipulation). We call this the *data independence objective*.

A second objective was to make the model structurally simple, so that all kinds of users and programmers could have a common understanding of the data, and could therefore communicate with one another about the database. We call this the *communicability objective*.

A third objective was to introduce high-level language concepts (but not specific syntax) to enable users to express operations upon large chunks of information at a time. This entailed providing a foundation for set-oriented processing (i.e., the ability to express in a single statement the processing of multiple sets of records at a time). We call this the *set-processing objective*.

There

* Source: "Relational Database: A Practical Foundation for Productivity", E. F. Codd, 1980 Turing Lecture

3.1 Relational Database: Definitions

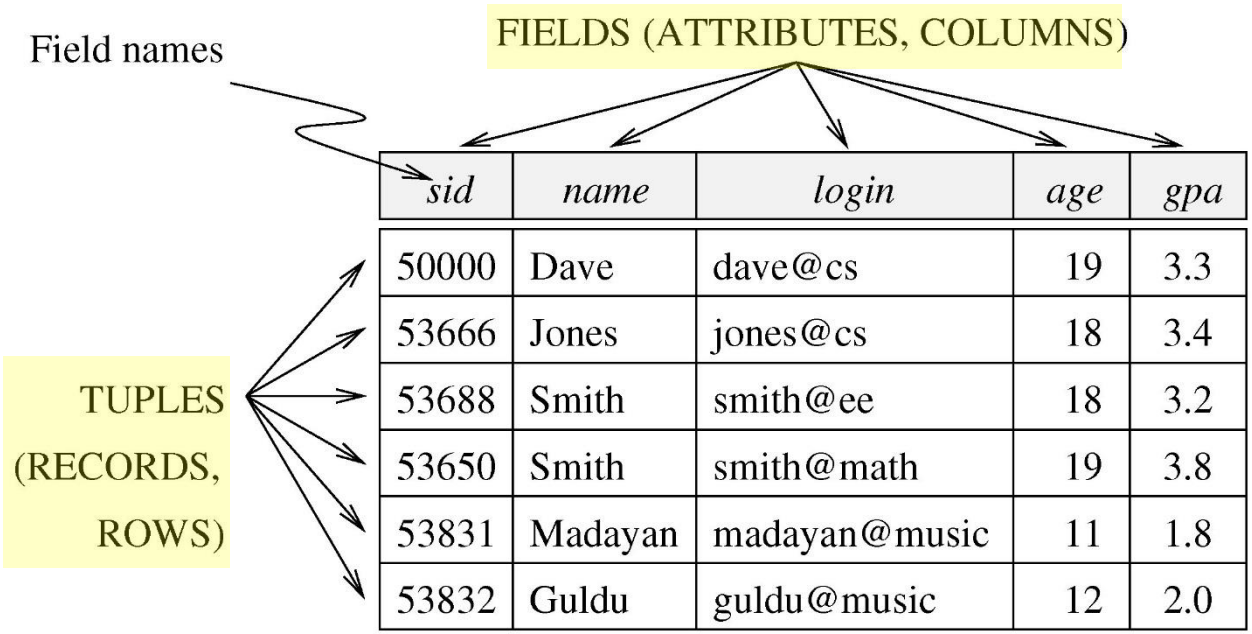
- Relational database: a set of relations
  - E.g. sample scott database: EMP, DEPT, SALGRADE

- A relation: made up of 2 parts:
  - Schema : relation name + name and type of each column.
    - ✓ e.g. students (*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).
  - Instance : a *table*, with rows and columns.
    - ✓ # of rows = *cardinality*, # of fields = *degree / arity*.

- A relation = a *set* of rows or *tuples* (i.e., all rows are distinct).

Student Relation: Example Instance

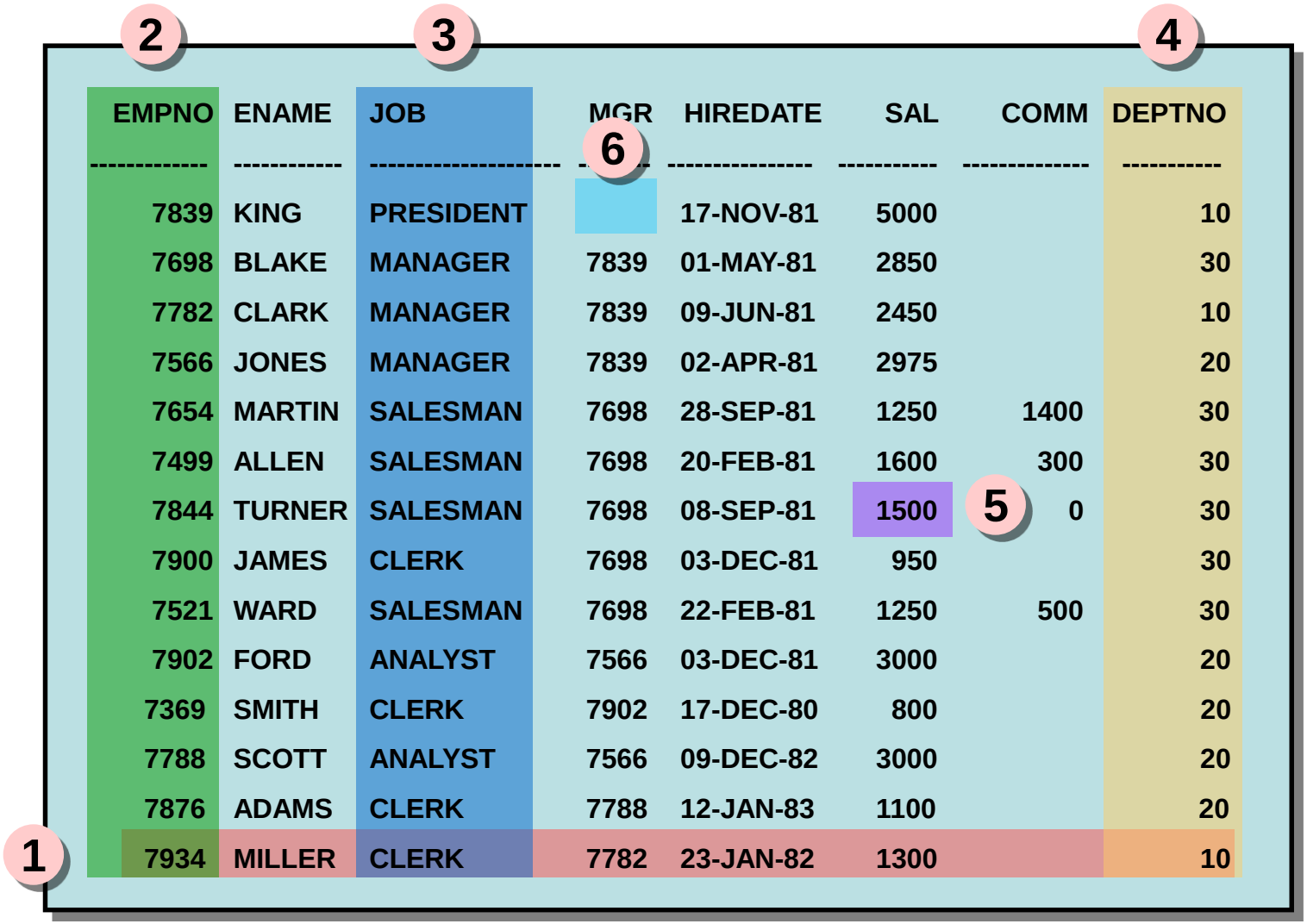
Students (*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real).



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

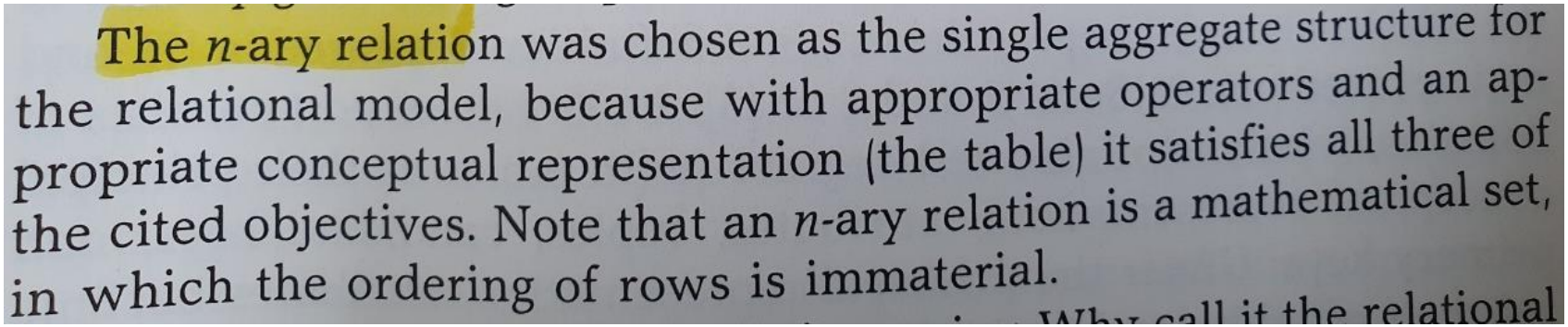
- *Relation or table*; attribute's domain or type; *domain constraints*
- Cardinality = 6, degree(or arity) = 5, all rows distinct
- Do all columns in a relation instance have to be distinct?
  - Key column(s): yes, other columns: no

Relation: Terminology



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

N-ary Relation ~~ Table



The *n*-ary relation was chosen as the single aggregate structure for the relational model, because with appropriate operators and an appropriate conceptual representation (the table) it satisfies all three of the cited objectives. Note that an *n*-ary relation is a mathematical set, in which the ordering of rows is immaterial.

* Source: "Relational Database: A Practical Foundation for Productivity", E. F. Codd, 1980 Turing Lecture

Why call RELATIONAL, not TABULAR?

- Based on the concept of "mathematical relation"
  - e.g. Order of tuples and order of columns do not matter!!
  - Data addressing is based on name and value, not on location
    - e.g. Select ename from emp where deptno = 30;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
.....							

    - c.f. **Excel** data model: (i,j)
    - Note: select …. order by empno; vs. order by 1;
- For better data independence!

Why call RELATIONAL, not TABULAR?

Sometimes the following questions arise: Why call it the relational model? Why not call it the tabular model? There are two reasons: (1) At the time the relational model was introduced, many people in data processing felt that a relation (or relationship) among two or more objects must be represented by a linked data structure (so the name was selected to counter this misconception); (2) tables are at a lower level of abstraction than relations, since they give the impression that positional (array-type) addressing is applicable (which is not true of *n*-ary relations), and they fail to show that the information content of a table is independent of row order. Nevertheless, even with these minor flaws, tables are the most important conceptual representation of relations, because they are universally understood.

considered as a serious alter-

* Source: "Relational Database: A Practical Foundation for Productivity", E. F. Codd, 1980 Turing Lecture

Creating Relations in SQL - DDL

- Students relation (to represent entity)
  - Type (domain) of each field is specified, and enforced by th
    added or modified.

```
CREATE TABLE Students
        (sid: CHAR(20),
         name: CHAR(20),
         login: CHAR(10),
         age: INTEGER,
         gpa: REAL)
```

- Enrolled table (to represent relationship)
  - holds information about courses that students take.

```
CREATE TABLE Enrolled
        (sid: CHAR(20),
         cid: CHAR(20),
         grade: CHAR(2))
```

- Other DDL example: view, index creation

```
CREATE TABLE Courses
        (cid: CHAR(20),
         cname: CHAR(20),
        …..)
```

- Data types
  - E.g. Oracle's data types (https://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm#CNCPT012)
  - More on Oracle date and time (http://infolab.stanford.edu/~ullman/fcdb/oracle/or-time.html)

Modifying Relations - DML

- Insert a new tuple into a table

```
INSERT INTO  Students (sid, name, login, age, gpa)
     VALUES  (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Delete all tuples satisfying some condition
  - e.g., name = 'Smith'

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

- Update a relation

```
UPDATE Students S
SET S.gpa = S.gpa – 0.1
WHERE S.gpa >= 3.3
```

Query Relations – DML(2)

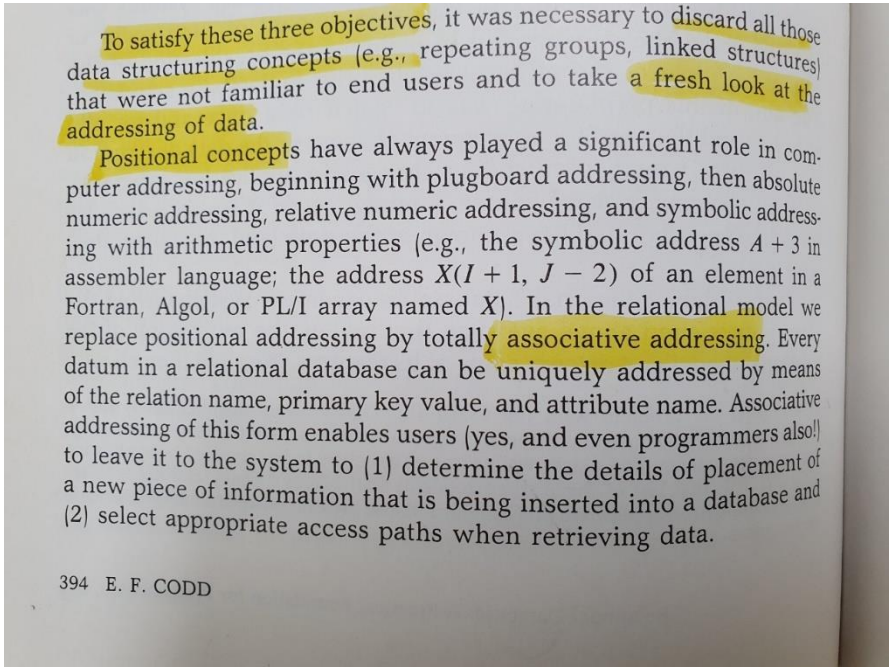
- Query relation(s)
  - Ch 3.6, Ch 4, Ch 5

```
SELECT *
FROM Students S
WHERE S.gpa >= 3.3  and  S.aga < 25;
```

*** SET-ORIENTED SEMANTICS!!**
vs. *record-at-a-time*

Addressing Data By Value

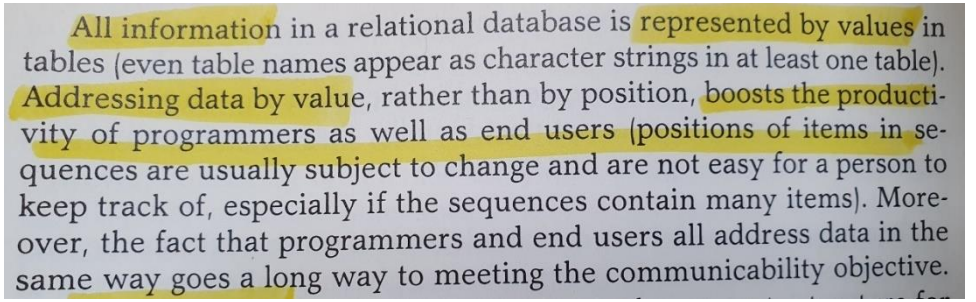
- Associative addressing (vs. positional concept)



To satisfy these three objectives, it was necessary to discard all those data structuring concepts (e.g., repeating groups, linked structures) that were not familiar to end users and to take a fresh look at the addressing of data.

Positional concepts have always played a significant role in computer addressing, beginning with plugboard addressing, then absolute numeric addressing, relative numeric addressing, and symbolic addressing with arithmetic properties (e.g., the symbolic address $A + 3$ in assembler language; the address $X(I + 1, J - 2)$ of an element in a Fortran, Algol, or PL/I array named X). In the relational model we replace positional addressing by totally associative addressing. Every datum in a relational database can be uniquely addressed by means of the relation name, primary key value, and attribute name. Associative addressing of this form enables users (yes, and even programmers also!) to leave it to the system to (1) determine the details of placement of a new piece of information that is being inserted into a database and (2) select appropriate access paths when retrieving data.

394 E. F. CODD



All information in a relational database is represented by values in tables (even table names appear as character strings in at least one table). Addressing data by value, rather than by position, boosts the productivity of programmers as well as end users (positions of items in sequences are usually subject to change and are not easy for a person to keep track of, especially if the sequences contain many items). Moreover, the fact that programmers and end users all address data in the same way goes a long way to meeting the communicability objective.

* Source: “Relational Database: A Practical Foundation for Productivity”, E. F. Codd, 1980 Turing Lecture

3.2 Integrity Constraints (ICs)

- IC: condition that must be true for any instance of the database.
  - E.g., domain constraint, primary/foreign key constraint
- Database should not allow illegal instances.
  - e.g., data entry errors
  - A legal instance of a relation satisfies all specified ICs.

- Where does ICs come from?
  - Database models a read-world: Entity and Relationship.
  - Read-world enterprises (e.g., Bank, University) have numerous / diverse policies, conditions, regulations about their entities or relationships between entities.
    - ✓ E.g., no minus balance in bank account; the minimum number of undergraduate course > 10
  - Such constraints should be correctly reflected also in databases.

Integrity Constraints (ICs)

- Who is responsible for checking and enforcing ICs?
  - Users, application programs, or DBMS?
  - **What if** ICs are managed by users or application logics?
    - ✓ Note that ICs will change constantly

- DBMS is the best place to manage ICs.
  - If ICs are checked and enforced automatically by DBMS, data are likely to be more faithful to real-world meaning.
  - Further, users/developers are relieved from the burden of coding/maintaining ICs using application logics.

Integrity Constraints (ICs)

- IC support in DBMS
  - Specified when schema is defined by users (e.g. DB designer or admin.)
  - Stored/managed by DBMS
  - Checked when relations are modified (e.g. insert/delete/update)
  - Error will be returned when an update violates any ICs.
- We can check a DB instance to see if an IC is violated, but we can NEVER infer that an IC is true by looking at an instance.
  - An IC is a statement about all possible instances, not only about current one!
  - e.g. the assertion that *sid* is a key to Student table is given by DB designer. We can never infer that sid is a key of Student relation simply looking at the current Student instance in Fig 3.4.
- Primary / foreign key ICs are the most common; more general ICs are supported too by RDBMSs. (section 3.2.3 skipped; revisited in section 5.7)

3.2.1 Primary Keys and Entity Integrity

- **A set of fields** is a KEY for a relation if :
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.

- Part 2 false? A SUPERKEY.

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

- E.g., *sid* is a key for Students. The set {*sid, gpa*} is a superkey. What about *name* column?

- If there's >1 key for a relation (CANDIDATE keys), one of the keys is chosen (by DBA) to be the PRIMARY key.

- Index on primary key (Index will be detailed in Ch 10.)

Primary and Candidate Keys in SQL

- Possibly many candidate keys (specified using UNIQUE), one of which is chosen as the primary key.
- Used carelessly, an IC can prevent the storage of database instances that arise in practice!
  - "For a given student and course, there is a single grade."

  vs.

  - "Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade."

```
CREATE TABLE Enrolled
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid) )
```

vs.

```
CREATE TABLE Enrolled
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid),
   UNIQUE (cid, grade) )
```

3.2.2 Foreign Keys & Referential Integrity

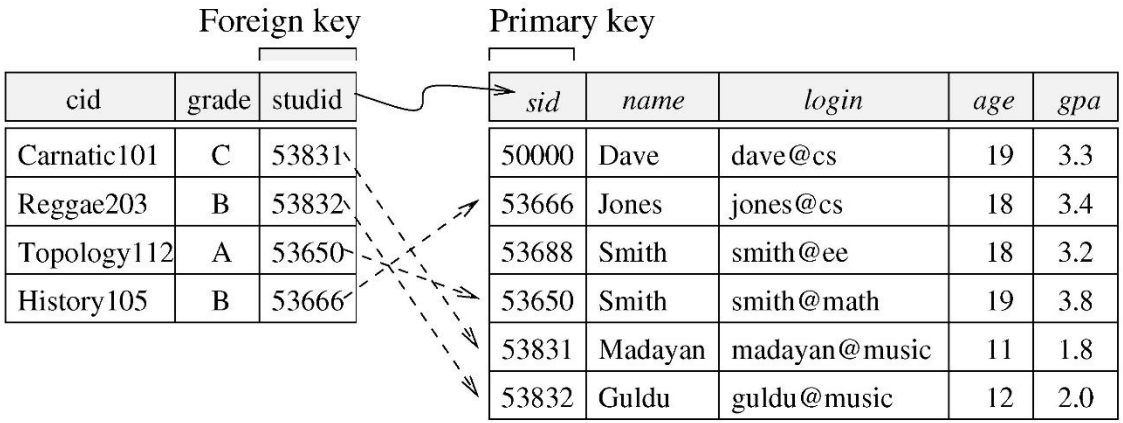
- Foreign key : Set of fields in one relation that is used to refer to a tuple in another relation.
  - Foreign key of the first relation must correspond to primary key of the second relation.)
  - It acks like a `logical pointer'.
  - Value-based relationship instance in relational data model
    - ✓ Helpful in providing better data independence.

- If all foreign key constraints are enforced, referential integrity is achieved, i.e., **no dangling references.**

Foreign Keys in SQL: Example

- Only students listed in the Students relation should be allowed to enroll for courses.

```
CREATE TABLE Enrolled
  (sid CHAR(20),  cid CHAR(20),  grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid) REFERENCES Students )
```



cid	grade	studid
Carnatic101	C	53831
Reggae203	B	53832
Topology112	A	53650
History105	B	53666

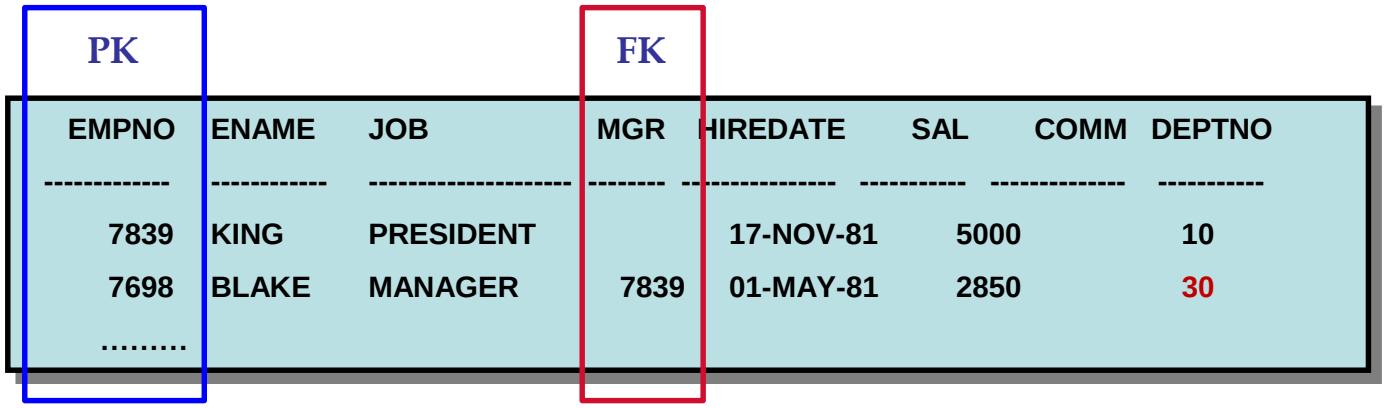
sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Child table — **Enrolled** (Referencing relation)

Students (Referenced relation) — **Parent table**

Foreign Key: Referencing The Same Table

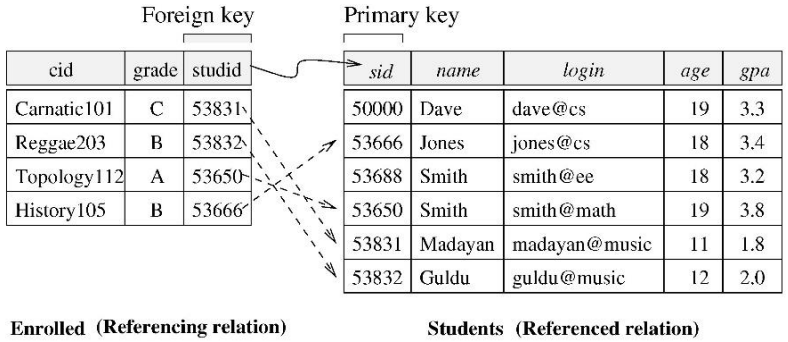
- Usually, FK in a relation refers to PK of other relation.
- But, FK could refer to the same relation; i.e., self-referencing




```
CREATE TABLE EMP
       (EMPNO NUMBER(4) PRIMARY KEY,
       ENAME VARCHAR2(10),
       JOB VARCHAR2(9),
       MGR NUMBER(4) REFERENCES EMP,
       ….
       DEPTNO NUMBER(2) CONSTRAINT FK_DEPTNO REFERENCES DEPT;
```

3.3 Enforcing Referential Integrity

- Consider Students and Enrolled;



- Insert/Delete tuples or Update FK in child table: What if an Enrolled tuple with a non-existent student id is inserted? Reject it for preventing dangling reference

- Insert/Delete tuples or Update PK in parent table:
  - What if a Students tuple is deleted?
    1. Delete all Enrolled tuples that refer to it. (CASCADE)
    2. Disallow deletion of a Students tuple that is referred to. (NO ACTION)
    3. (In SQL, also: Set sid in Enrolled tuples that refer to it to a special value null, denoting `unknown' or `inapplicable'.) (SET NULL)
    4. Set sid in Enrolled tuples that refer to it to a default sid. (SET DEFAULT)
  - When primary key of Students tuple is updated, four options similarly to the deletion case .

Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is NO ACTION (*delete/update is rejected*)
  - CASCADE (also delete all tuples that refer to deleted tuple)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
     ON DELETE CASCADE
     ON UPDATE SET DEFAULT)
```

- Oracle: no action/cascade/set null
  - "on delete cascade" (O)
  - "on update cascade" (X)
- MySQL:

Auto-creation of Index on primary key!
What about index on foreign key?

https://docs.oracle.com/cd/E17952_01/mysql-5.5-en/create-table-foreign-keys.html

What if IC is not properly enforced?

- Even some banks do not enforce global $0 invariants, as shown by a recent concurrent withdrawal attack on ATMs that extracted $40 million from only 12 account numbers. (CACM 2014 May)
  - **In Hours, Thieves Took $45M in A.T.M. Scheme (New York Times)**

IC and Transactions

- Within a transaction, an IC can be checked in either IMMEDIATE or DEFERRED mode.
  - Immediate mode (default)
    - ✓ checked at the end of a SQL statement
  - Deferred mode
    - ✓ SET CONSTRAINT **Foo** DEFERRED
    - ✓ Checked at transaction commit time

```
Begin_TX
Select ..
Insert into EMP ..
Insert into DEPT ..
...
End_TX
```

3.4 Relational Query Languages: SQL

- SQL : Structured Query Language
  - Pronounced as “ess-cue-ell” or “sequel”

- A major strength of the relational model
  - SQL as data sublanguage (cf. hierarchical or network data model)
  - Simple and powerful querying of data.

SQL: Standard Query Lang. for RDBMSs

- Based on Relational Algebra and Calculus (by Dr. Codd, Ch4)
- Developed by IBM System R team in the 1970s (PI: Dr. Don Chamberlin)
  - "Sequel: A Structured English Query Language", SIGFIDET 1974

- Need for a standard since it is used by many vendors
- Standards:
  - SQL-86: Originally IBM, then ANSI in 1986
  - SQL-89 (minor revision)
  - SQL-92 (major revision) – Basic subset; most systems support at least this
  - SQL-99 (major rev.) – "Object-Relational" concept (not fully supported yet)
  - SQL-2003 (major rev.) XML, window functions, sequences, ..(no full support yet)
  - 2008 added x-query stuff, new triggers,…; SQL:2011 ..
  - SQL:2016: the current standard (https://en.wikipedia.org/wiki/SQL:2016)

Querying Single Relation

- To find all 18 year old students, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
....
```

Relational Query Lang.: What vs. How

- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries. (WHAT!)
  - Allows the optimizer to extensively re-order operations, and still ensure the same result. (HOW!)
    - ✓ For the given query (WHAT), there exist many HOWs which return the same answer.
    - ✓ Among many HOWs, optimizer chooses the best one.
  - Separation of WHAT and HOW: another source of **data independence**
  - Easy writing and efficient evaluation of queries will boost **application development productivity**.

Querying Multiple Relations

- What does the following query compute? c.f. Excel does not support join

```
SELECT  S.name, E.cid
FROM  Students S, Enrolled E
WHERE  S.sid=E.sid AND E.grade="A"
```

Join query: An Example

Input: Relation(s)
Output: Relation

Given the following instance of Student and Enrolled

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
50000	Dave	dave@cs	19	3.3

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

we get:

S.name	E.cid
Smith	Topology112

Query specifies just **WHAT** I want;
User needs not care HOW?
➔ Declarative language
(vs. Procedural language)

DBMS query optimizer
translates WHAT into (best) HOW,
and query processor will evaluate the HOW.
(Ch 12, 14, 15)

3.5 Logical DB Design: ER → Relational

- We will revisit this issue later in DB design part.

3.6 Views

- A view is just a relation, but we store just a definition, rather than a set of tuples. (= virtual relation)

```
CREATE VIEW HighScoreStudents(sid, gpa)
    AS SELECT S.sid, S.gpa
       FROM Students S
       WHERE S.gpa > 3.5
```

HighStudent: **View**

sid	gpa
53650	3.8

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Student: **Base Relation**

Query on View

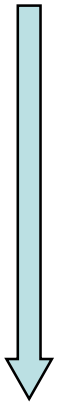
```
SELECT  S.sid
FROM  HighStudents S
```

sid	gpa
53650	3.8

HighStudent: View

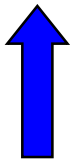
sid
53650

Result: as relation



Query Rewrite
(automatically by DBMS)

```
CREATE  VIEW  HighStudents(sid, gpa)
        AS  SELECT  S.sid, S.gpa
        FROM  Students S
        WHERE  S.gpa > 3.5
```



```
SELECT  S.sid
FROM  Students S
WHERE  S.gpa > 3.5
```

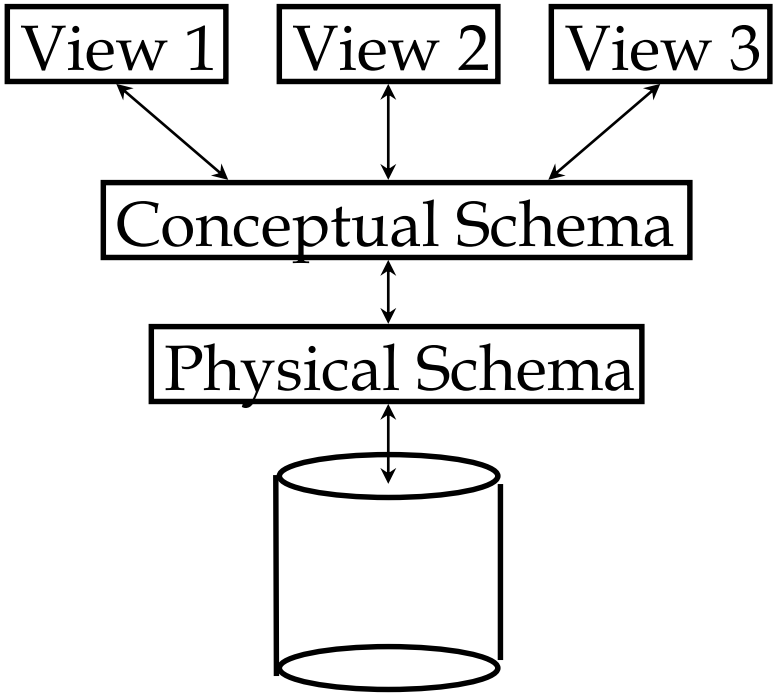
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Student: Base Relation

Views and Data Independence

- External schema can be defined using "views"

- Thus, the view mechanism provides "logical data independence"



Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - E.g. Allowing some users to access HighStudent view instead of Students, we can hide other students or other information such as name, login, and age. in.

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Student: **Base Relation**

HighStudent: **View**

sid	gpa
53650	3.8

Updates on Views

- View is also just another relation.
  - Thus, ideally every updates(insert/delete/update) should be allowed
  - BUT, certain types of views are not updatable.

```
CREATE VIEW HighStudents(sid, gpa)
       AS SELECT S.sid, S.gpa
          FROM Students S
          WHERE S.gpa > 3.5
```

sid	gpa
53650	3.8

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

3.7 Destroying and Altering Relations

DROP TABLE Students [RESTRICT | CASCADE]

- Destroys the relation Students.
  - Both schema and instance
  - cf. truncate command in Oracle

- Views can be dropped using the *DROP VIEW* command.
  - How to handle *DROP TABLE* if there's a view on the base table?
    - ✓ *DROP TABLE* command has options to let the user specify this.

ALTER TABLE Students ADD COLUMN firstYear: integer

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field. -> schema change/evolution

Relational Model: Summary

- Structure: A tabular representation of data.
- Operation: Powerful and natural query languages exist.
  - SQL: Structured Query Language - "sequel"
  - In addition, Insert/Delete/Update operation
- Constraints: Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we *always* have domain constraints.
- Simple, easy-to-learn and intuitive; currently the most widely used.