Outline

Lecture topics:

- why study and use database systems?
- central concepts of database systems
- brief introduction to standard SQL

References:

- text 3rd edition: Chapter 1; Chapter 2; Chapter 8, sections 1-5
- text 4th edition: Chapter 1; Chapter 2; Chapter 8, sections 1-6; Chapter 9, section 2; Chapter 24

Applications of database technology

Original:

- inventory control
- payroll
- electronic funds transfer
- reservations systems

More recent:

- computer aided design (CAD)
- computer aided software engineering (CASE), development environments
- geographic/environmental information systems (GIS, EIS)
- telecommunications systems
- information systems (“IS”) of all kinds

Common principles

- data is formatted
- data is important:
  - need to remember data reliably
  - need to manipulate data easily
- there are large amounts of data:
  - need to use mass store
- many users require simultaneous access to data:
  - need concurrency control
- many diverse applications access the data:
  - need security

- need **data independence**

Data management

Basic idea:

- remove details related to data storage and access from application programs
- concentrate those functions in single subsystem: the **Database Management System** (DBMS)
- have all applications access data through the DBMS
- make applications **independent** of data storage

...continued

Advantages:

- uncontrolled redundancy can be reduced
- less risk of inconsistency
- data integrity can be maintained
- access restrictions can be applied
- conflicting requirements can be balanced

But most importantly:

- a higher degree of **data independence** can be achieved

Definitions

- *Database*: a collection of related files containing data
  - structure of the files is described in a *data dictionary*, which itself is stored in the database
- *Database Management System* (DBMS): a collection of programs that manipulate a database, for example to:
  - set up the storage structures
  - load data (contents of database)
  - perform updates to data
  - process queries (requests for data retrieval) from applications and users
- The DBMS provides a central point of access to the data

Program–data independence

Objective:

- to isolate application programs as much as possible from changes to:
  - data
  - descriptions of data

Two kinds of data independence:

- **physical data independence** (application programs immune to changes in storage structures)
- **logical data independence** (application programs immune to changes to descriptions of unrelated data)

...continued

Examples of changes in storage structures:

- data encoding
- record structure
- file structure

Data dependence is expensive because changes in the way data is stored or described requires changes in application programs.

Brief history of data management

First generation (50's and 60's), files on tape:

- batch processing
- sequential files on tape
- input on punched cards
- growing application base

Second generation (60's), files on disk:

- disks enabled random access files
- new access methods (ISAM, hash files) were developed
- mostly batch with some interactive processing
- independent application systems with separate files
- growing application base

...continued

As application base grows:

- many shared files
- a multitude of file structures
- a need to exchange data between applications

Variety of problems:

- redundancy: multiple copies
- inconsistency: independent updates
- inaccuracy: concurrent update mishandled
- incompatibility: multiple formats, constraints
- insecurity: proliferation
- not auditable: poor chain of responsibility
- inflexibility: changes difficult to apply

...continued

Third generation (mid 60's and 70's), early database systems:

- beginning to separate between logical view and physical implementation
- network model and hierarchical model introduced
- first batch oriented; on-line support added later
- transaction management added (concurrency control, recovery)
- access control facilities provided

...continued

Fourth generation (80's), relational technology:

- simple, solid conceptual model
- strict separation of logical view and physical implementation
- powerful, set-oriented query languages (SQL)
- distributed databases emerging

Fifth generation (now), post-relational systems:

- added functionality, more complex data (temporal, spatial)
- object-oriented systems
- serving a broader class of applications
  - logic-based deductive systems
  - active databases
  - multidatabase systems

The “three-schema” architecture

schema (or scheme): description of data contents, structure, and possibly other aspects of a database

external schema (view): describes data as seen by an application program or by an end user

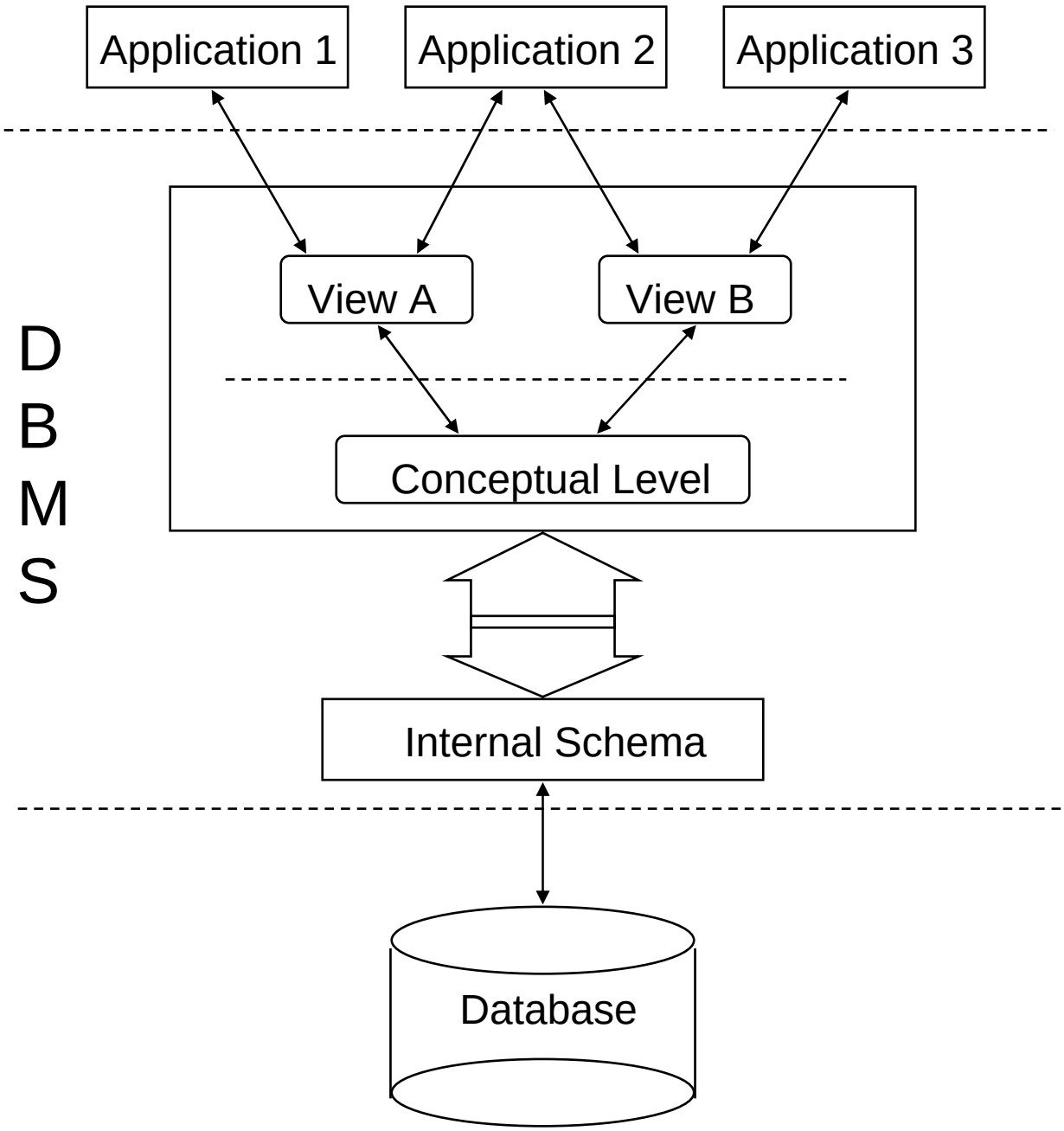
conceptual schema: describes the base logical structure of all data

internal schema: describes how the database is physically encoded, including selection of files, indexes, etc.

...continued

- Separation of external schema from conceptual schema enables logical data independence
- Separation of conceptual schema from internal schema enables physical data independence
- Database schema (**intention**) is different from database instance (**extension**)

...continued


Application 1
Application 2
Application 3
View A
View B
Conceptual Level
DBMS
Internal Schema
Database

Example schemas

- External: programming-language specifications:
  - C:
    - typedef struct {
      char emp_no[6];
      float salary; } empl;
  - Pascal:
    - type employee = record
      EmployeeNum : array[1..6] of char;
      Department : integer; end;
- Conceptual:

```
employee
   number character(6)
   salary floating_point
   department integer
```

- Internal:

```
emp_record length=19; nopad;
   empnum ds x'7';
   empsal ds 1d;
   empdept ds 1f;
```

Interface to the DBMS

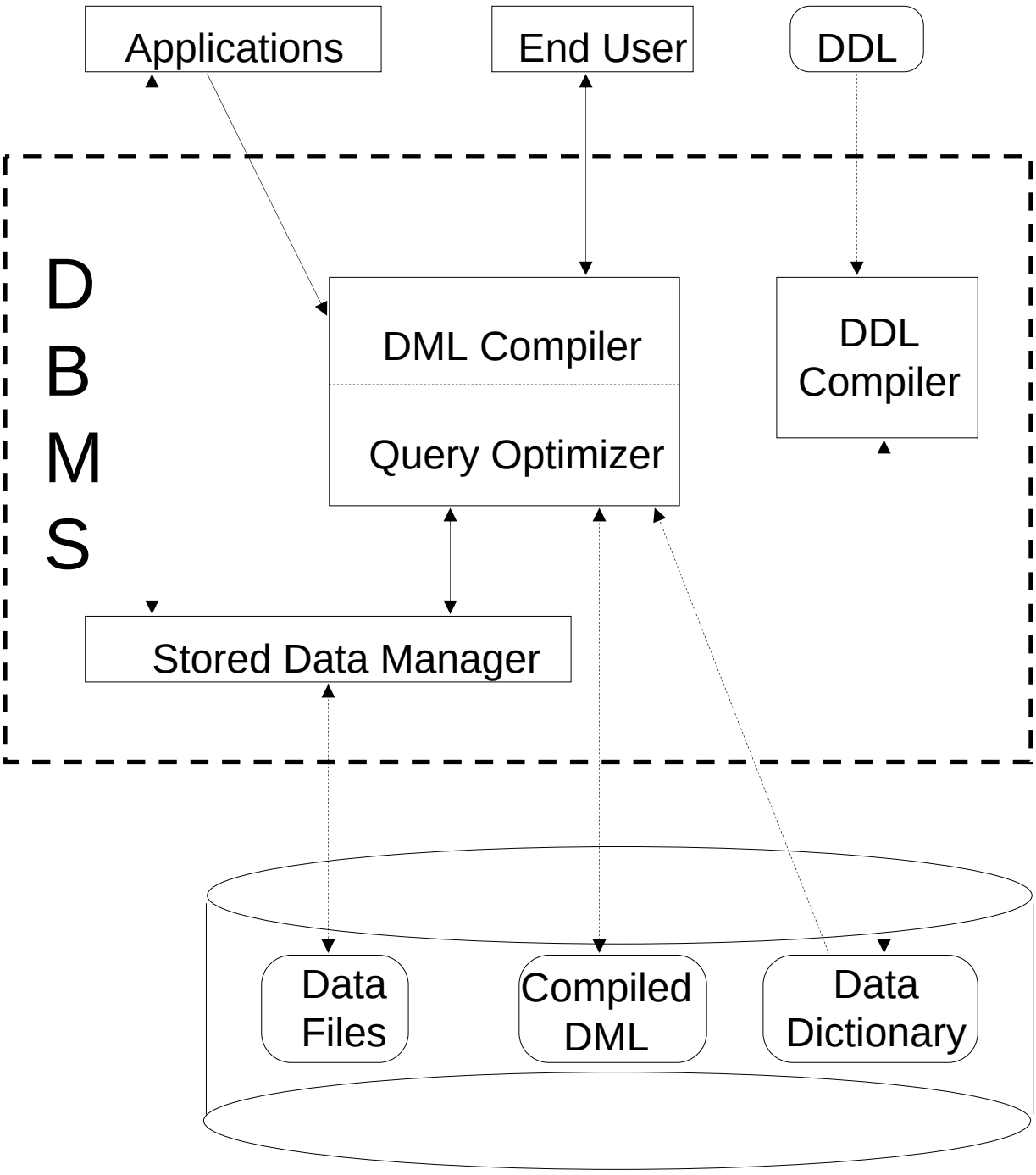
Data Definition Language (DDL)

- for specifying schemas
- may have different DDLs for external, conceptual and internal schemas
- information is stored in the *data dictionary*

Data Manipulation Language (DML)

- for specifying data queries and updates
- two general ways of querying and updating a database
  - through “stand alone” DML facilities
  - from within application programs
- two kinds of DMLs
  - **navigational** (one record at a time)
  - **non-navigational**

Components of a DBMS



Requirements for a DBMS

- provide data definition facilities
  - define a data definition language (DDL)
  - provide a user-accessible catalogue (data dictionary) (database should be self-describing)
- provide facilities for storing, retrieving and updating data
  - define a data manipulation language (DML)
- support multiple views of data (user views)
  - end user or application should see only the data needed, and in form required

...continued

- provide facilities for specifying constraints on contents:
  - primary key constraints (identity integrity)
  - foreign key constraints (referential integrity)
  - more general constraints
- provide facilities for controlling access to data:
  - prevent unauthorized access and update
- allow simultaneous access and update by multiple users:
  - provide a concurrency control mechanism

...continued

- support (logical) transactions:
  - a sequence of operations to be performed as an atomic action
  - all operations are performed or none
  - equivalent to performing the operations instantaneously
- provide facilities for database recovery:
  - must never lose the database, for whatever reason
  - bring the database back to a consistent state after a failure (disk failure, faulty program, earthquake, and so on)
- provide facilities for database maintenance (utilities):
  - maintenance operations: redefine, unload, reload, mass insertion and deletion, validation, reorganization,... (preferably without needing to shut down system)

Concurrency

- Operations of different users may interleave. The final result may not be correct unless special precautions are taken. This is called the **concurrent update problem:**

	User A	**User B**
1.	Read AMT (10)	Read AMT (10)
2.	Decremented by 5	Decremented by 3
3.	Write AMT	Write AMT

(i) Read AMT into local variable AMT1 (for A)

(ii) Read AMT into local variable AMT2 (for B)

(iii) Decrement AMT1 by 5

(iv) Decrement AMT2 by 3

(v) Write AMT from AMT1 (for A)

(vi) Write AMT from AMT2 (for B)

AMT = 7 ???

Security

- Security is the protection of data against unauthorized disclosure, alteration or destruction
  - physical security
    - protecting the physical resources from fire, flood,…
    - control of physical access to the system
  - against unauthorized users
    - accomplished by username–password
    - some users may not be permitted to see all data (access control) – restrictions defined when a username is assigned

Data Integrity

- Data integrity is meant to ensure that the data in the database are accurate and meaningful
  - Data have properties, e.g.:

    Employee(Emp#, Name, Dept, Salary);
    Manager(EMP#, Dept)

    might have constraints like:

    Salary < 1,000,000;
    Emp# is unique

Types of users

end user

- naive: accesses DBMS through applications
- sophisticated: writes ad-hoc queries using DML

application developer

- (implementation) Implements end-user applications to access the database
  - using traditional programming languages with DML libraries or embedded DML
  - using application generators
- (analysis) Develops application specifications
  - using DDL to define application views
  - using CASE tool

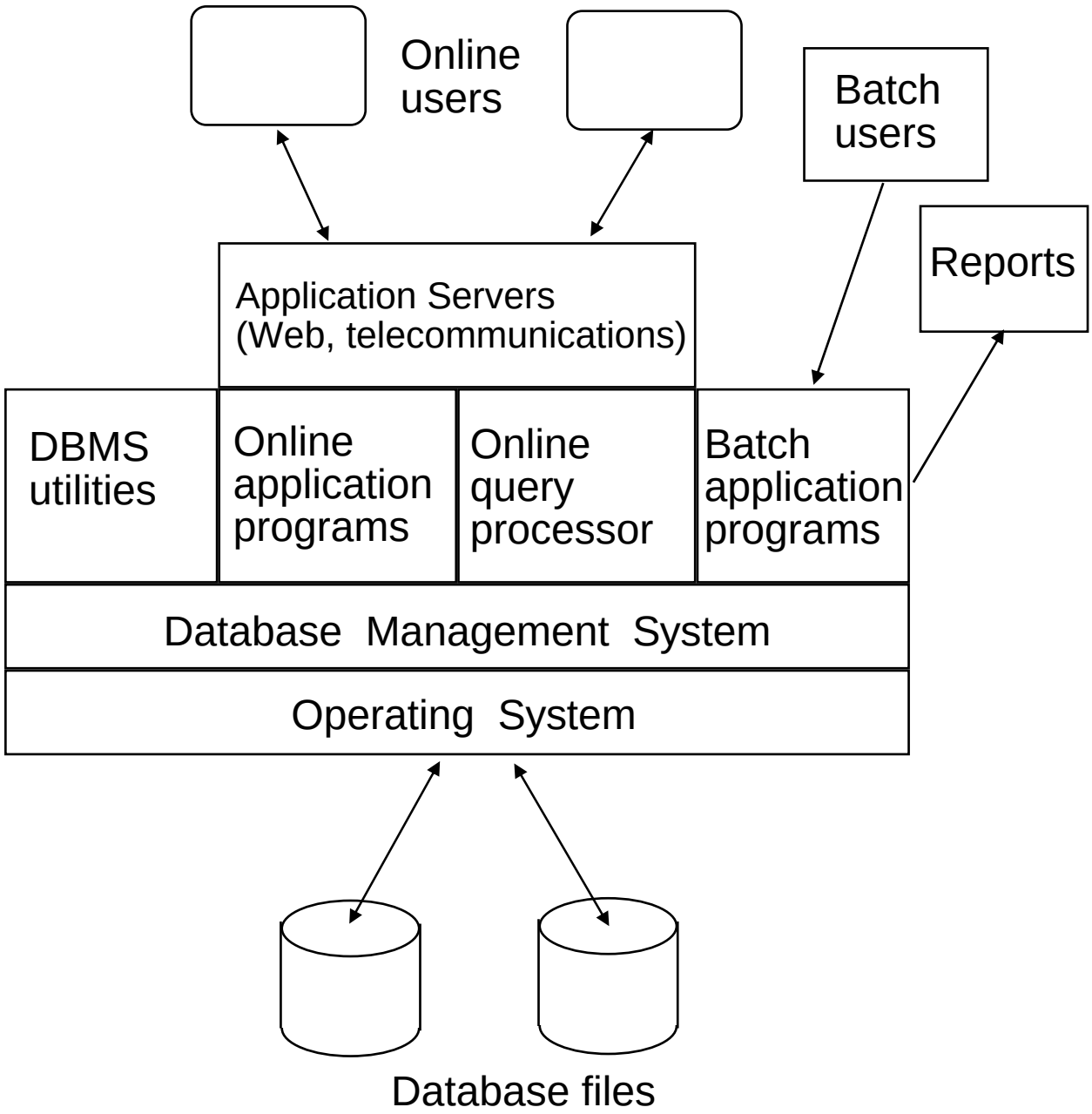
database administrator (DBA)

database system implementor

Role of a database administrator (DBA)

- manages conceptual schema
- assists with application view integration
- monitors overall performance of DBMS
- defines internal schema
- loads and reformats database
- is responsible for security and reliability

Typical DB Environment



…continued

- *Applications server*: the interface between interactive users and system
  - routes, formats messages
  - provides communication infrastructure
    - Web servers
    - distributed systems (dialup, private networks)
- *Application programs*: perform specific functions e.g. payroll, inventory, etc.
- *DBMS utilities*: tools for DBA to manage the DBMS
  - load/unload, backup
  - data reorganization
  - analysis tools for performance
- *Operating system*: resource manager for computer system
  - manage memory, access I/O devices

Overview of SQL

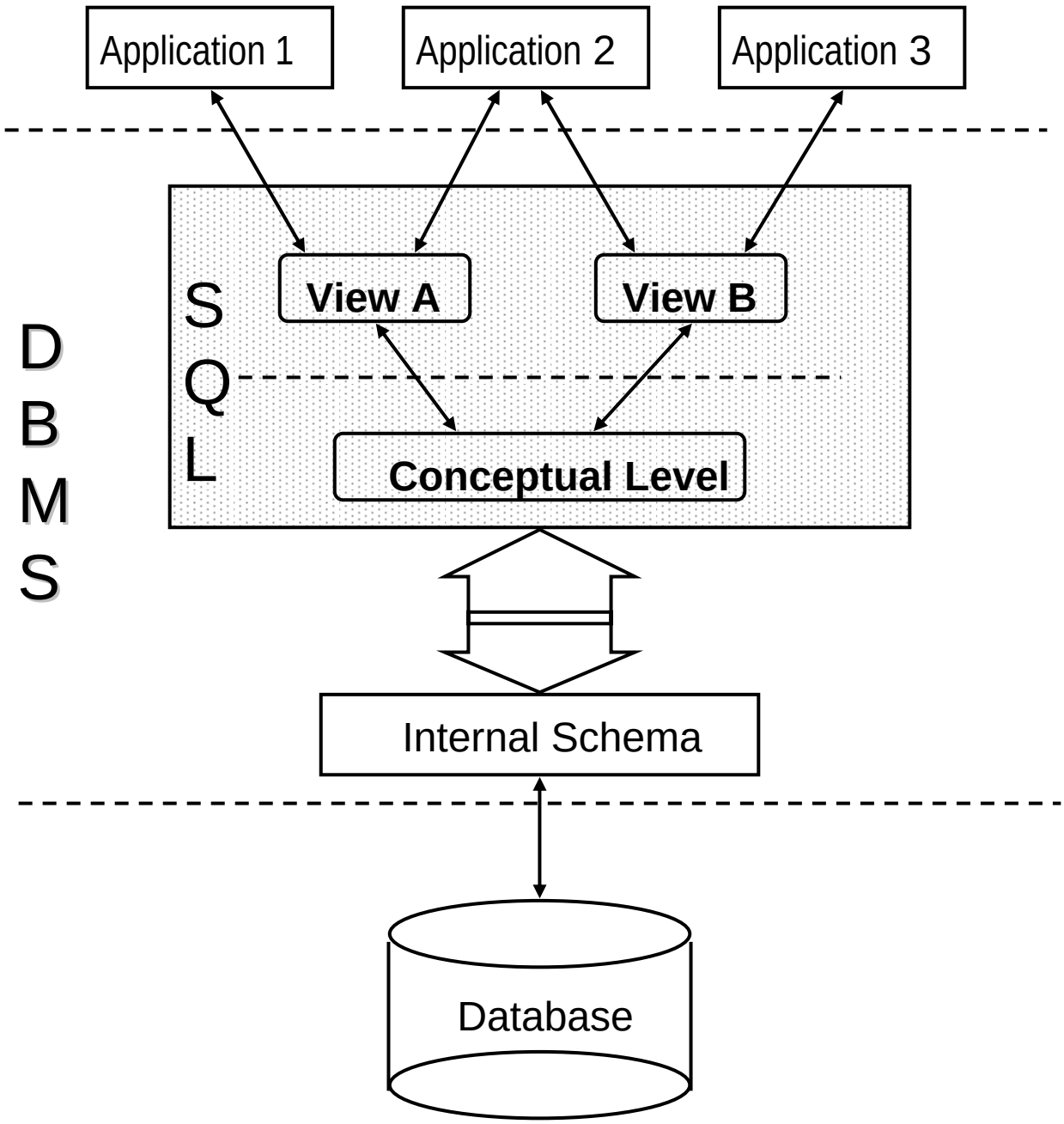
- Structured Query Language (SQL, sometimes pronounced “sequel”)
- ISO 9075, an international standard for relational database systems
- the standard is evolving:
  - 1986: SQL1; initial version, 75 pages
  - 1989: SQL89; most commercial products conform to this version, with extensions
  - 1992: SQL2; three levels of conformity, ~600 pages
  - 1999: SQL99 (sometimes called SQL3), ~1200 pages

...continued

Main features:

- powerful view definition language
- integrity constraints in conceptual schema
- DML can be embedded in various programming languages (called “embedded SQL”), or used via programming libraries
  - object/class libraries in OO environments
- transaction control
- authorization sublanguage/model

...continued


Application 1
Application 2
Application 3
DBMS
SQL
View A
View B
Conceptual Level
Internal Schema
Database

Underlying relational model

Example relational database for a credit card company

Vendor

Vno	Vname	City	Vbal
1	Sears	Toronto	200.00
2	Walmart	Ottawa	671.05
3	Esso	Montreal	0.00
4	Esso	Waterloo	2.25

Customer

AccNum	Cname	Prov	Cbal	Climit
101	Smith	Ont	25.15	2000
102	Jones	BC	2014.00	2500
103	Martin	Que	150.00	1000

Transaction

Tno	Vno	AccNum	Tdate	Amount
1001	2	101	20060115	13.25
1002	2	103	20060116	19.00
1003	3	101	20060115	25.00
1004	4	102	20060120	16.13
1005	4	103	20060125	33.12

Structure of a relational database

Database: collection of uniquely named **tables** (**relations**)

Relation: set of **rows** (**tuples**)

Attribute: column

Domain: set of allowed values for an attribute

Attribute values must be **atomic** (single values): no tuples or sets or repetition

Row: distinguishable thing

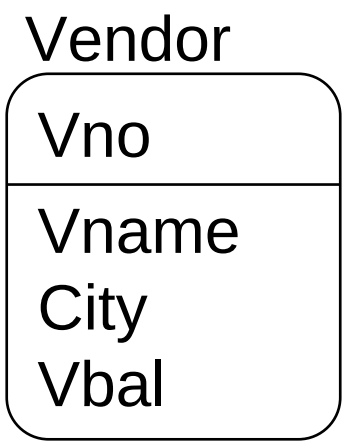
Table: set of related things

Diagrammatic conventions

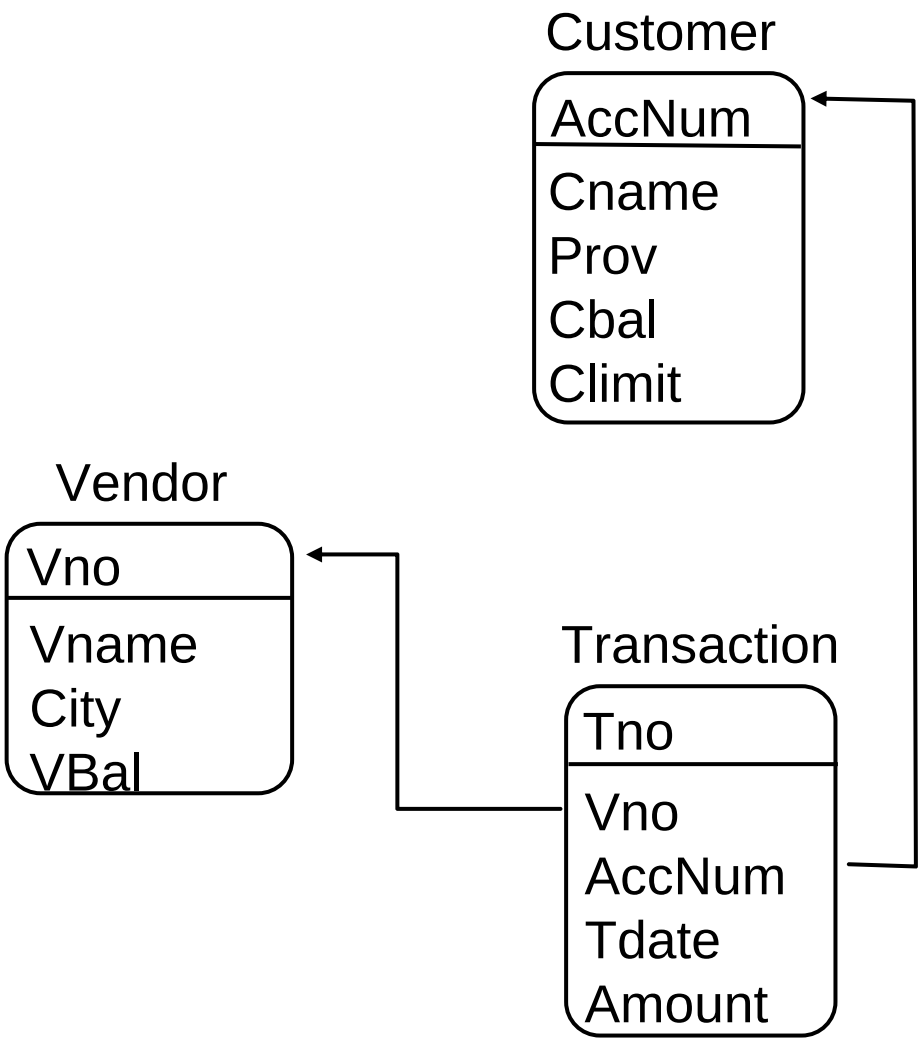
Vendor

Vno	Vname	City	Vbal

or



Pictorial schema



The SQL DDL

- used for defining
  - tables
  - views
- example of table definition (conceptual schema):

```
create table Vendor
(Vno      INTEGER not null,
 Vname    VARCHAR(20),
 City     VARCHAR(10),
 Vbal     DECIMAL(10,2),
 primary key (Vno) );
```

...continued

```
create table Customer
(AccNum INTEGER not null,
 Cname  VARCHAR(20) not null,
 Prov   VARCHAR(20),
 Cbal   DECIMAL(6,2) not null,
 Climit DECIMAL(4,0) not null,
 primary key (AccNum) );

create table Transaction
(Tno    INTEGER not null,
 Vno    INTEGER not null,
 AccNum INTEGER not null,
 Tdate  DATE,
 Amount DECIMAL(6,2) not null,
 primary key (Tno),
 foreign key (Vno)
   references vendor(Vno),
 foreign key (AccNum)
   references Customer(AccNum) );
```

Attribute domains in SQL

- **INTEGER**: integers representable with 32 bits
- **SMALLINT**: integers representable with 16 bits
- **DECIMAL(m,n)**: fixed point numbers
- **FLOAT**: 32 bit floating point numbers
- **CHAR(n)**: fixed length strings
- **VARCHAR(n)**: variable length strings
- **BIT(n)**: n bits
- **BIT VARYING(n)**: variable number of bits

...continued

- **DATE** (year, month, day)
- **TIME** (hour, minute, second)
- **TIME(i)** (hour, minute, second, second fraction)
- **TIMESTAMP** (date, time, second fraction)
- **INTERVAL YEAR/MONTH** (year month interval)
- **INTERVAL DAY/TIME** (day time interval)

- plus many, many product-specific (non-standard) extensions

Modifying table definitions

- Table schemas can be changed after the table has been created:
  - adding columns
  - removing columns
  - removing constraints (e.g. p-key)
  - some SQL implementations allow
    - renaming a column
    - modifying a column
- Example:

```
ALTER TABLE Vendor
ADD Street VARCHAR(15)
```

Removing tables

- SQL operation is “drop”
- Tables can be dropped at any time
- Dropping a table deletes the schema and the instance
- All views, foreign-key definitions are also removed
- Example:
  `DROP TABLE Transaction`

The SQL DML

SQL has a *non-navigational* DML:

E.g. “Find names and provinces of customers who owe more than $1000 to the company.”

```
select Cname, Prov
  from Customer
  where Cbal > 1000;
```

...continued

- basic querying:

```
select columns
  from R1,…, Rk
  [where filter ];
```

- result is a relation over *columns* (*columns* = “*” means all attributes in R_1,..., R_k)
- R_1,..., R_k: tables from which the data is retrieved
- *filter* : conditions on tuples used to form the result; optional

...continued

- conditions may include:
  - arithmetic operators +, -, *, /
  - comparisons =, <>, <, <=, >, >=
  - logical connectives **and**, **or** and **not**

E.g. “List the names of the customers who live in Ontario and whose balance is over 80% of their balance limit.”

```
select Cname
  from Customer
  where Prov = 'Ont' and
        Cbal > 0.8 * Climit;
```

...continued

- basic insertion:

```
insert into Customer
values (104, 'Trevor', 'ON',
0, 4000);
```

- deletion: delete Customer rows for customers named Smith:

```
delete from Customer
where Cname = 'Smith';
```

- delete all transactions:

```
delete from Transaction;
```

...continued

- modification, changing existing rows
- set balance of account 102 to zero:

```
update Customer set Cbal = 0
where AccNum = 102;
```

- add $100 to each customer's monthly limit:

```
update Customer set Climit =
Climit + 100;
```

SQL external schema

- called **views**
- a view is a named query (result is usually computed when the view is used)

```
create view WatVendors as
  select VNo, VName, VBal
  from Vendor
  where City = 'Waterloo';
```

- views can be used in retrieval exactly like tables (but updates of views are restricted)

Advantages of views

- logical data independence
- simplified perception of the database
- different views for different users
- restricting data access