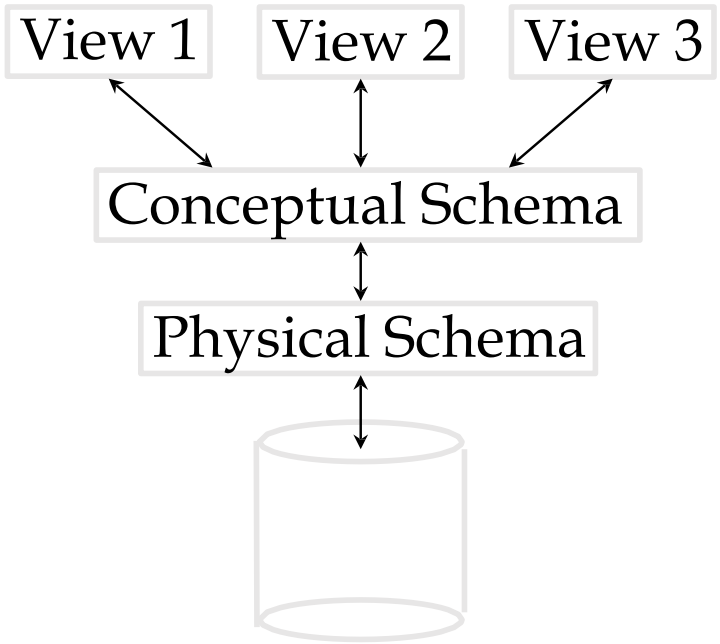
Data Models

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using the a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



☛ *Schemas are defined using DDL; data is modified/queried using DML.*

Example: University Database

- Conceptual schema:
- *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*

- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.

- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence *

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

☛ *One of the most important benefits of using a DBMS!*

Transaction?

As services are employed for serious purposes, they will inevitably update information resources

- Can we be sure that such updates preserve integrity?
- What about when multiple services need to work together?
- What about when the services are involved in a long-lived activity?

Transactions: 1

- A transaction is a computation (i.e., program *in execution*) that accesses and possibly modifies a database:
  - Not the source code; not the binaries
  - Can be interleaved with other transactions
  - But guarantees certain correctness properties

The purpose of the transaction concept is to avoid the problems ("race conditions") that may arise from interleaving

ACID Properties

- If programmers guarantee correctness of individual transactions, then DBMS guarantees correctness of any set of them
- The ACID properties formalize the notion of a transaction behaving as one operation
  - (Failure) *Atomicity*—all or none—if failed then no changes to DB or messages
    - This is the vernacular notion of “transaction”
  - *Consistency*—don't violate DB integrity constraints: execution of the op is correct
  - *Isolation* (Atomicity)—partial results are hidden
  - *Durability*—effects (of transactions that "happened" or committed) are forever

Transaction Lifecycle

A transaction goes through well-defined stages in its life (always *terminating*)

- Inactive
- Active (may read and write)
  - *Entire* business logic takes place here
- Precommit (no errors during execution; needed for mutual commitment protocols)
- Failed (errors)
- Committed (the DBMS decides this)
- Forgotten (the DBMS reclaims data structures)

Schedules

- Schedules are histories of computations showing all events of interest
- Schedule of $T_1 \ldots T_n$ has all ops of $T_1 \ldots T_n$ in the same order as within each T_i, but *interleaved* across T_i to model concurrency
  - Includes active transactions
  - Typically a partial order among events
- Two challenges
  - What are the bad schedules?
  - How can the DBMS prevent them?

Conflict

Order-sensitivity of operations

- Two operations of different transactions, but on the same data item, *conflict* if
  - Their mutual order is significant, i.e., determines at least one of the following:
    - The final value of that item read by future transactions
    - The value of the item as read by present transactions

Serial Schedules

Transactions are wholly before or after others (i.e., occur one by one)

- Clearly, we must allow for service requests to come in slowly, one-by-one
- Thus, under independence of transactions (assuming each transaction is correct), serial schedules are obviously correct

Serializable Schedules

- Interleaved schedules are desirable
  - Why?
- Those *equivalent* to some serial schedule. Here equivalent can mean
  - *Conflict* equivalent—all pairs of conflicting ops are ordered the same way
  - *View* equivalent—all users get the same view

Locks

Lock item *x* while using item *x*

- Binary: at most one party may lock *x*
  - Lock(x): acquire the lock
    - Computation hangs until lock(x) returns, i.e., the lock is acquired
  - Unlock(x): relinquish the lock
  - Gives mutual exclusion but restrictive

Multimode Locks

When one party has an exclusive lock, no other party may have an exclusive or a shared lock

- Shared-lock(x) needed for read(x)
  - Others can also hold a shared lock
- Exclusive-lock(x) needed for write(x)
  - No one else can concurrently hold a shared or exclusive lock
- Can be upgraded (read to write) or downgraded (write to read)

Locking

- By itself, using locks does not guarantee serializability
  - What is an example of a bad schedule obtained while using locks?
- A locking protocol, i.e., how locks are acquired and released, is critical
- That is, locks on different data items must be related

The Log

- The following actions are recorded in the log:
  - *Ti writes an object*: The old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *Ti commits/aborts*: A log record indicating this action.
- Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- Log is often *duplexed* and *archived* on "stable" storage.
- All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

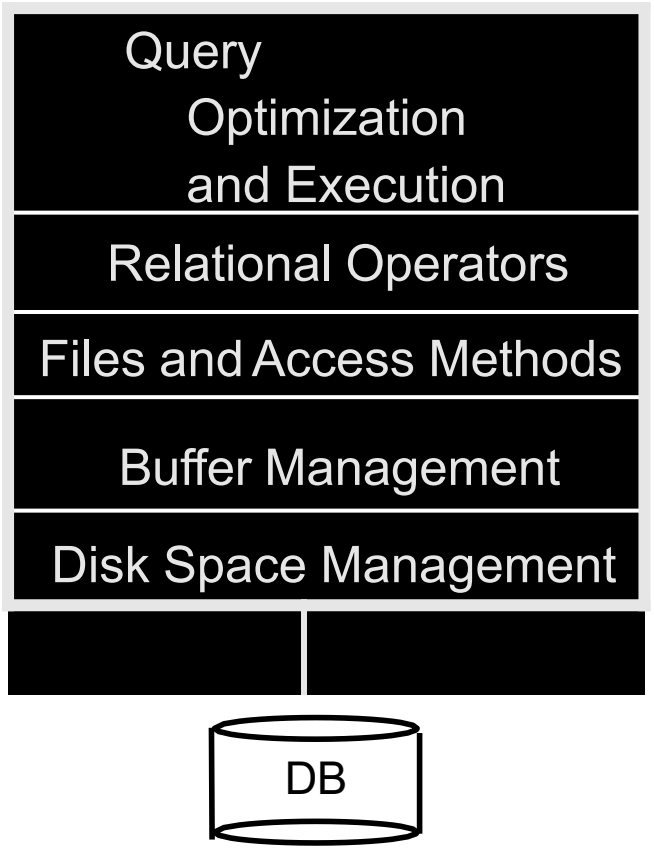
Databases make these folks happy

- End users and DBMS vendors
- DB application programmers
  - E.g., smart webmasters
- *Database administrator (DBA)*
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

Must understand how a DBMS works!

Structure of a DBMS

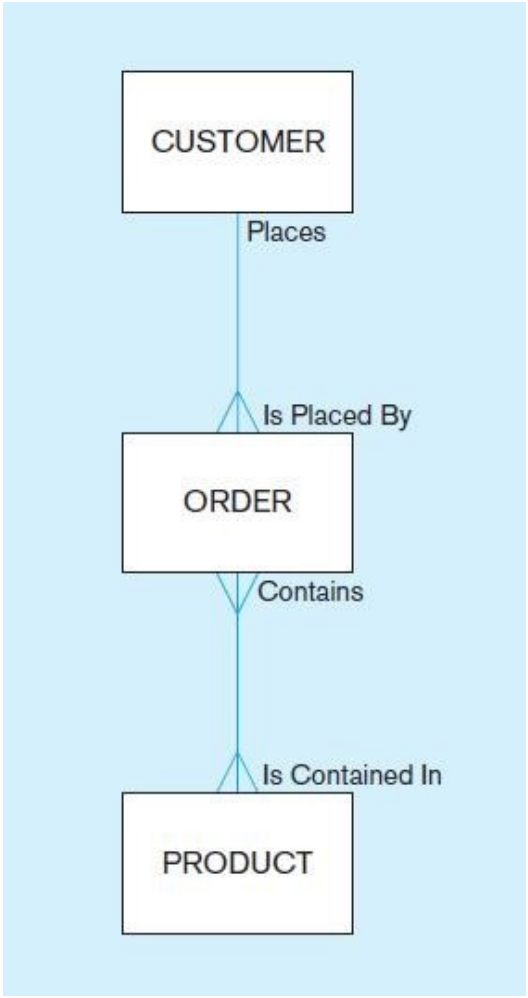
- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.



ELEMENTS OF THE DATABASE APPROACH

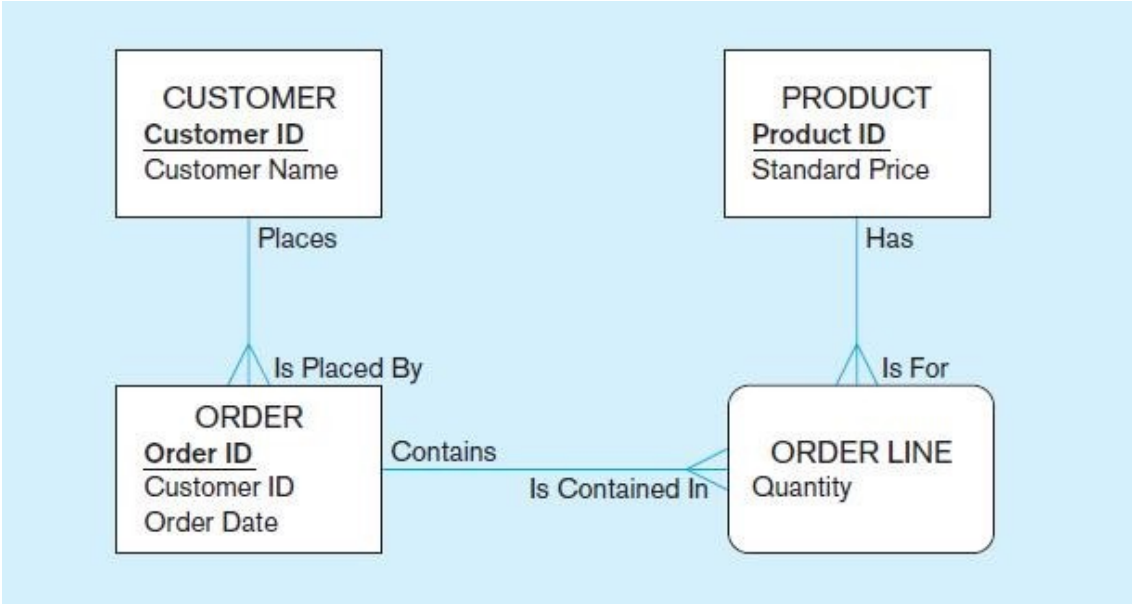
- Data models
  - Graphical diagram capturing nature and relationship of data
  - Enterprise Data Model–high-level entities and relationships for the organization
  - Project Data Model–more detailed view, matching data structure in database or data warehouse
- Entities
  - Noun form describing a person, place, object, event, or concept
  - Composed of attributes
- Relationships
  - Between entities
  - Usually one-to-many (1:M) or many-to-many (M:N), but could also be one-to-one (1:1)
- Relational Databases
  - Database technology involving tables (relations) representing entities and primary/foreign keys representing relationships

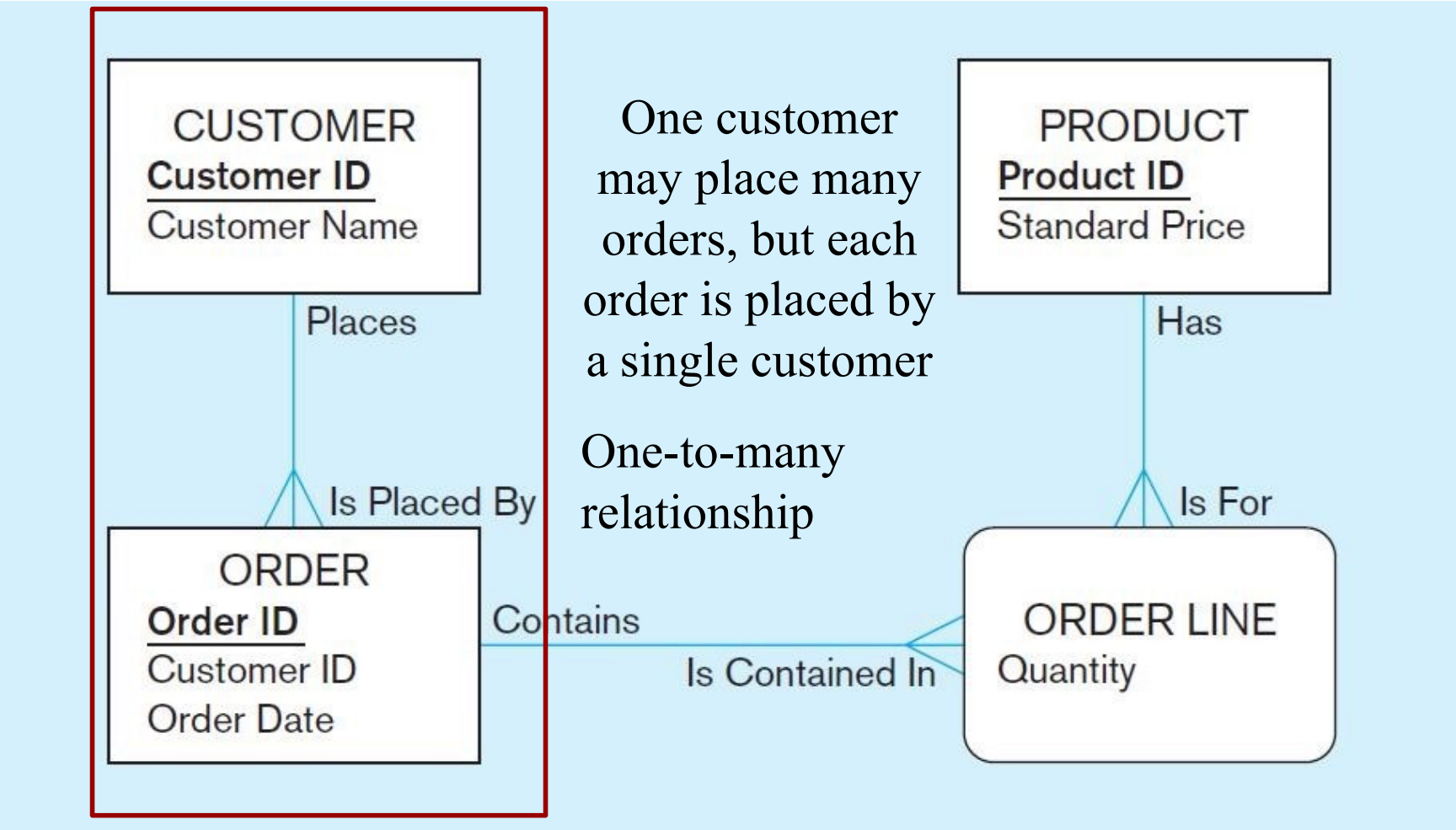
Figure 1-3 Comparison of enterprise and project level data models



Segment of an enterprise data model

Segment of a project-level





RDBMS and MySQL

Introducing Relational Databases

- A relational database manages data in tables.
- Databases are managed by a relational database management system (RDBMS).
- An RDBMS supports a database language to create and delete databases and to manage and search data.
- The database language used in almost all DBMSs is SQL.

Introducing Relational Databases

- After creating a database, the most common SQL statements used are
  - INSERT to add data
  - UPDATE to change data
  - DELETE to remove data
  - SELECT to search data
- A database table may have multiple columns, or attributes, each of which has a name.
- Tables usually have a primary key, which is one or more values that uniquely identify each row in a table

Relational Databases

Winery Table

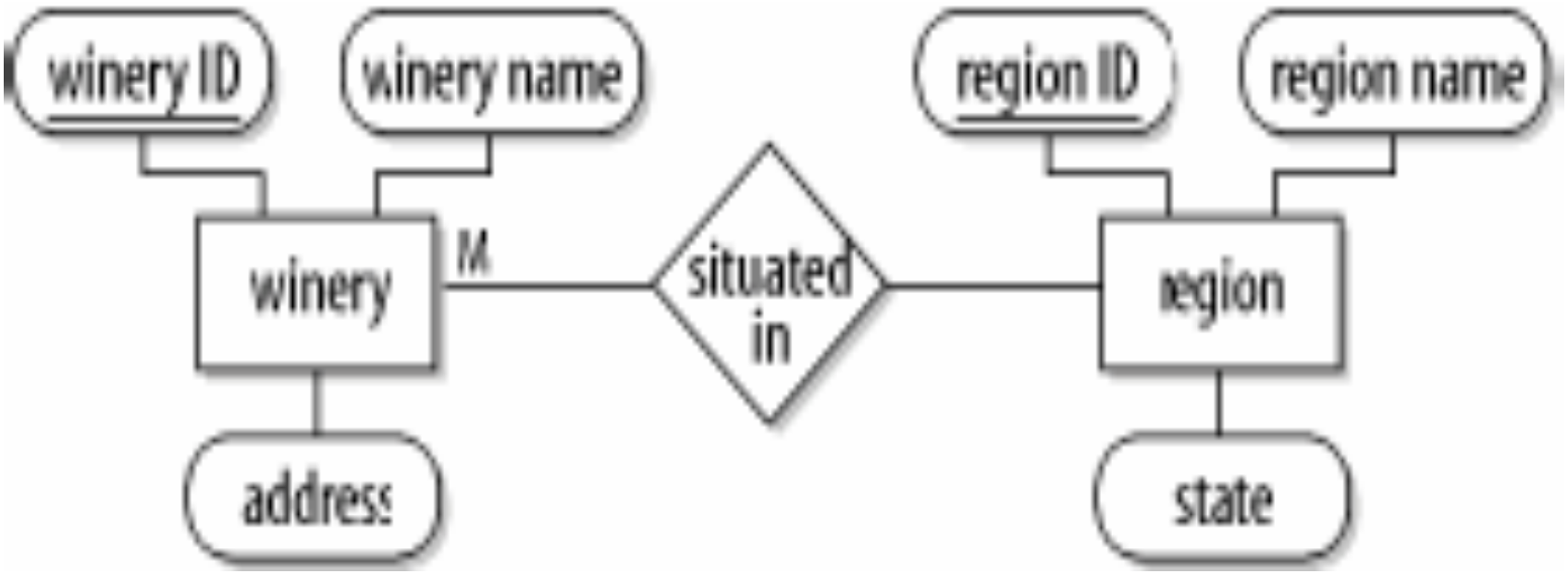
Winery ID	Winery name	Address	Region ID
1	Moss Brothers	Smith Rd.	3
2	Hardy Brothers	Jones St.	1
3	Penfolds	Arthurton Rd.	1
4	Lindemans	Smith Ave.	2
5	Orlando	Jones St.	1

Region Table

Region ID	Region name	State
1	Barossa Valley	South Australia
2	Yarra Valley	Victoria
3	Margaret River	Western Australia

Relational Databases

- A database is modeled using entity-relationship (ER) modeling.



Terminology

- Database
  - A repository to store data.
- Table
  - The part of a database that stores the data. A table has columns or attributes, and the data stored in rows.
- Attributes
  - The columns in a table. All rows in table entities have the same attributes. For example, a customer table might have the attributes name, address, and city. Each attribute has a data type such as string, integer, or date.

Terminology

- Rows
  - The data entries in a table. Rows contain values for each attribute. For example, a row in a customer table might contain the values "Matthew Richardson," "Punt Road," and "Richmond." Rows are also known as records.
- Relational model
  - A model that uses tables to store data and manage the relationship between tables.
- Relational database management system
  - A software system that manages data in a database and is based on the relational model.

Terminology

- SQL
  - A query language that interacts with a DBMS. SQL is a set of statements to manage databases, tables, and data.
- Constraints
  - Restrictions or limitations on tables and attributes. For example, a wine can be produced only by one winery, an order for wine can't exist if it isn't associated with a customer, having a name attribute could be mandatory for a customer.

Terminology

- Primary key
  - One or more attributes that contain values that uniquely identify each row. For example, a customer table might have the primary key of cust ID. The cust ID attribute is then assigned a unique value for each customer. A primary key is a constraint of most tables.
- Index
  - A data structure used for fast access to rows in a table. An index is usually built for the primary key of each table and can then be used to quickly find a particular row. Indexes are also defined and built for other attributes when those attributes are frequently used in queries.

Terminology

- Entity-relationship modeling
  - A technique used to describe the real-world data in terms of entities, attributes, and relationships.
- Normalized database
  - A correctly designed database that is created from an ER model. There are different types or levels of normalization, and a third-normal form database is generally regarded as being an acceptably designed relational database.

Managing Databases

- The Data Definition Language (DDL) is the set of SQL statements used to manage a database.

MySQL Overview

MySQL Server

- Community Server
- Enterprise Server
- Embedded Server
- Cluster (Standard and Carrier-Grade)

MySQL GUI Tools

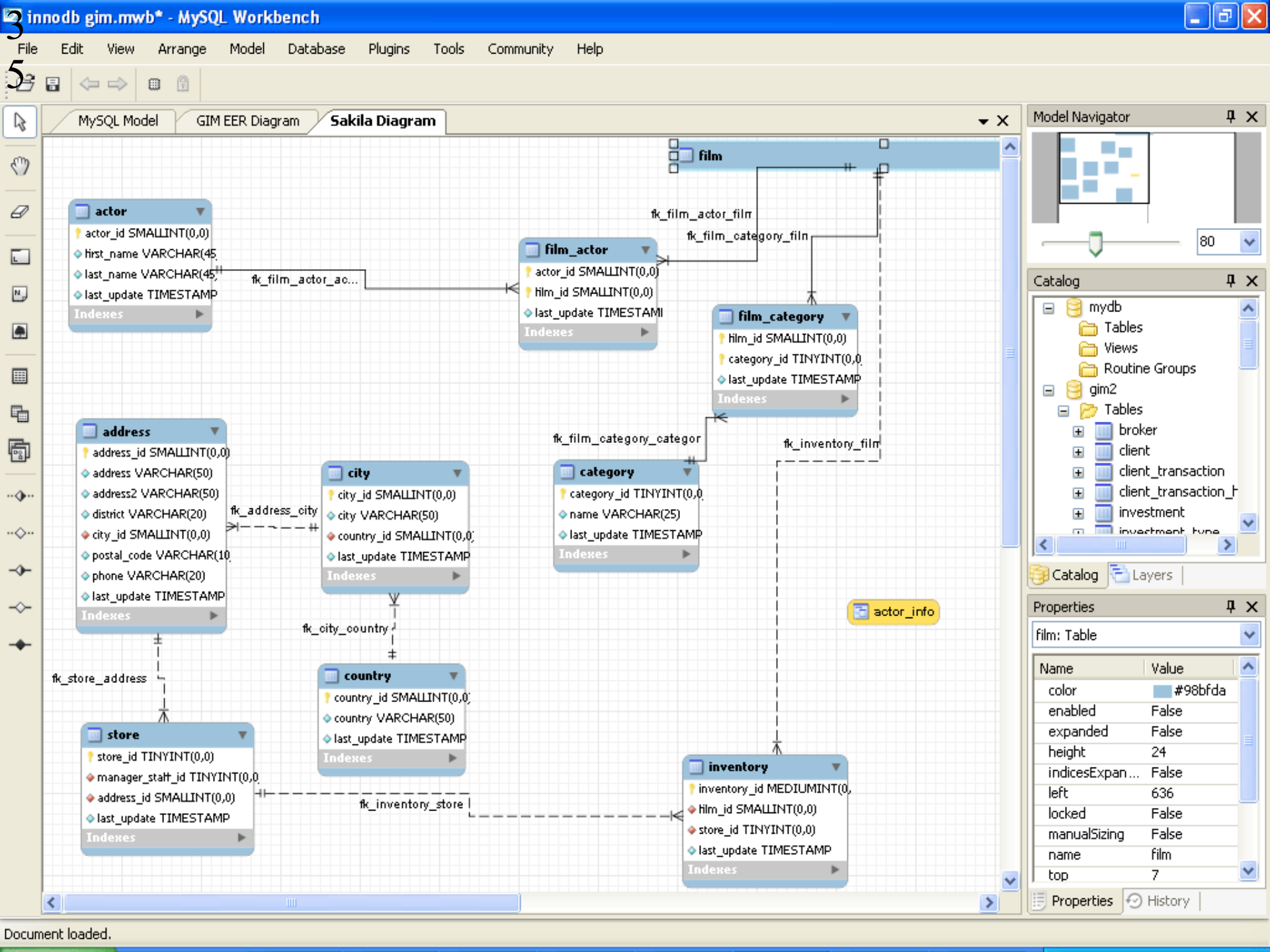
- Query Browser
- Administrator
- Migration Toolkit
- Visual Studio Plug-in
- MySQL Workbench

MySQL Drivers

- JDBC
- ODBC
- .NET
- PHP

MySQL Workbench

- MySQL Workbench enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.


innodb gim.mwb* - MySQL Workbench
File Edit View Arrange Model Database Plugins Tools Community Help
MySQL Model
GIM EER Diagram
Sakila Diagram
film
actor
actor_id SMALLINT(0,0)
first_name VARCHAR(45
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
fk_film_actor_ac...
film_actor
actor_id SMALLINT(0,0)
film_id SMALLINT(0,0)
last_update TIMESTAMI
Indexes
fk_film_actor_film
fk_film_category_film
film_category
film_id SMALLINT(0,0)
category_id TINYINT(0,0
last_update TIMESTAMP
Indexes
fk_film_category_categor
category
category_id TINYINT(0,0
name VARCHAR(25)
last_update TIMESTAMP
Indexes
fk_inventory_film
address
address_id SMALLINT(0,0)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(0,0)
postal_code VARCHAR(10
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
fk_address_city
city
city_id SMALLINT(0,0)
city VARCHAR(50)
country_id SMALLINT(0,0
last_update TIMESTAMP
Indexes
fk_city_country
country
country_id SMALLINT(0,0
country VARCHAR(50)
last_update TIMESTAMP
Indexes
fk_store_address
store
store_id TINYINT(0,0)
manager_staff_id TINYINT(0,0
address_id SMALLINT(0,0)
last_update TIMESTAMP
Indexes
fk_inventory_store
inventory
inventory_id MEDIUMINT(0,
film_id SMALLINT(0,0)
store_id TINYINT(0,0)
last_update TIMESTAMP
Indexes
actor_info
Model Navigator
80
Catalog
mydb
Tables
Views
Routine Groups
gim2
Tables
broker
client
client_transaction
client_transaction_h
investment
Catalog
Layers
Properties
film: Table
Name
Value
color
#98bfda
enabled
False
expanded
False
height
24
indicesExpan...
False
left
636
locked
False
manualSizing
False
name
film
top
7
Properties
History
Document loaded.

MySQL Workbench

- Characteristics

❖ **Forward and Reverse Engineering**

- A visual data model can easily be transformed into a
- physical database on a target MySQL Server with just a few mouse clicks.
- it can also import SQL scripts to build models and export models to DDL scripts that can be run at a later time.

❖ **Change Management**

Sakila Sample Database