Final Review

Lecture 13

Format

- 7-8 problems, with multiple sub-parts
- No notes, calculators, books, computers, phones, etc. may be used
- Your responses must be written in pen

Content

Everything, including…

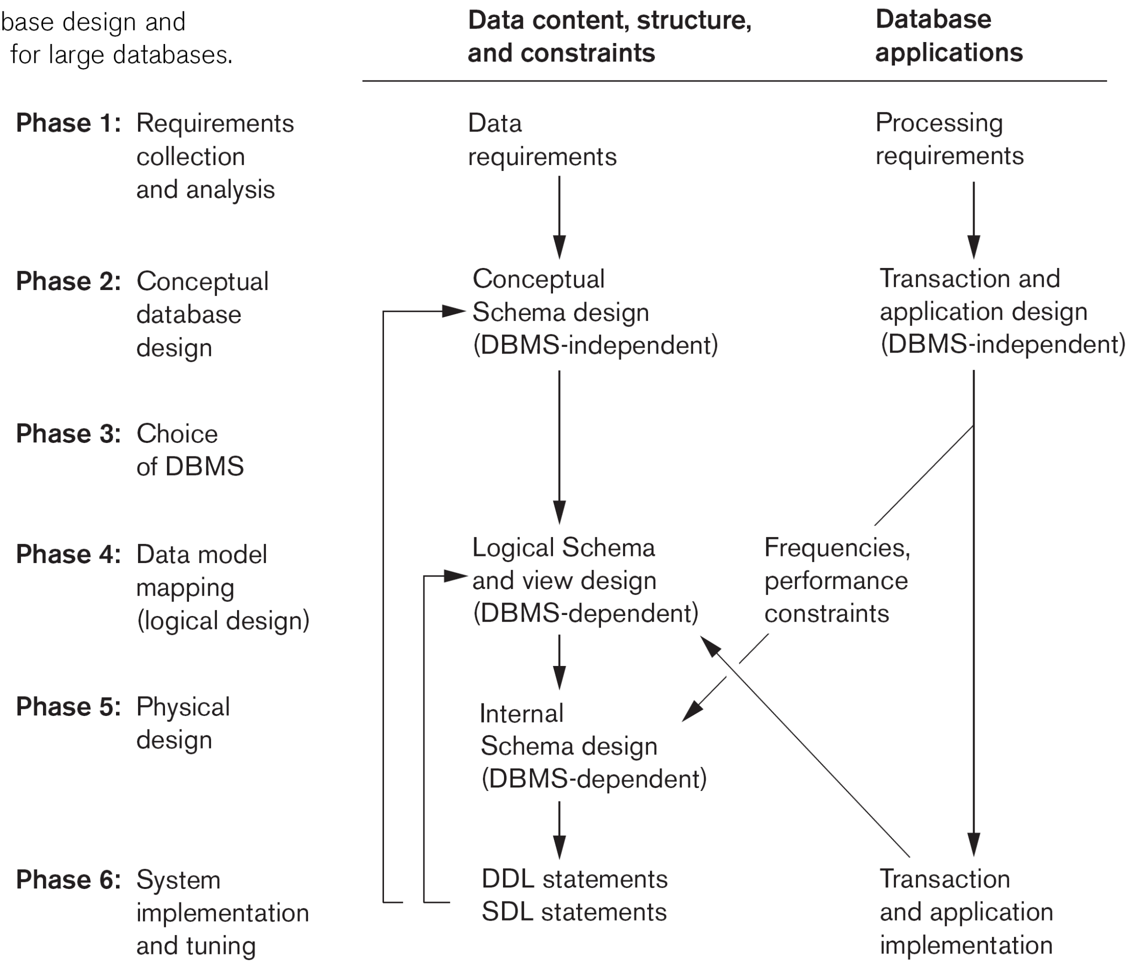
- General database knowledge
- ER diagrams
- The relational model
- Mapping ER diagrams to relations
- Normalization
- SQL [programming]
- Physical design/tuning
- Database security
- Inverted index

General Database Knowledge

- What is a transaction?
  - What are the properties that should hold for effective transaction processing?
- What is SQL?
- What are the major phases of database design?
  - How do these apply to the material we've discussed: ERDs, relations, normalization, denormalization, indexes, views, SQL

Database Design & Implementation

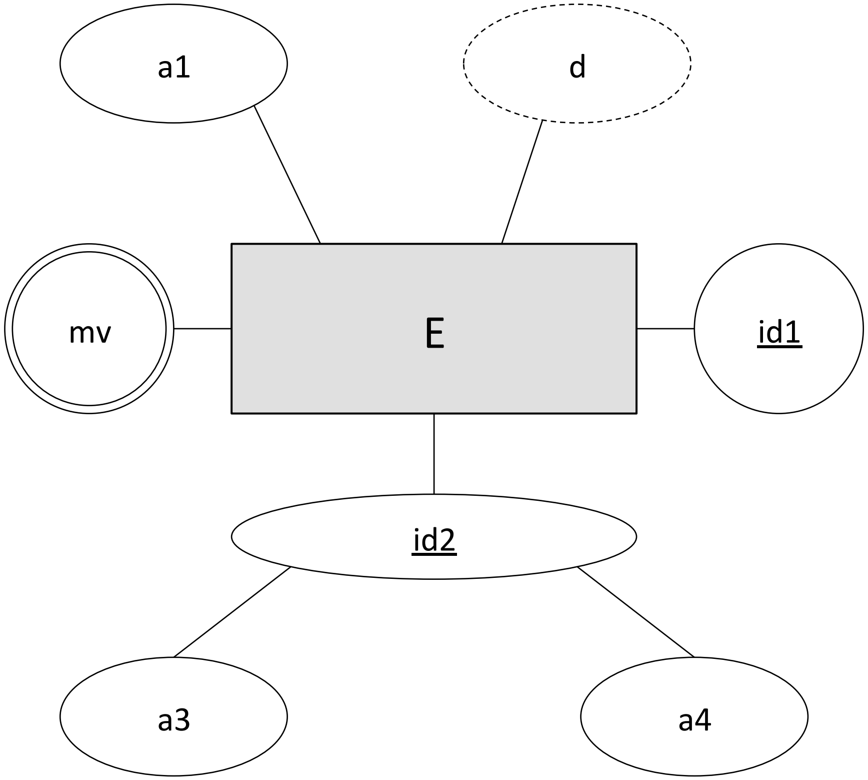
Figure 10.1
Phases of database design and implementation for large databases.



	Data content, structure, and constraints		**Database applications**
Phase 1: Requirements collection and analysis	Data requirements		Processing requirements
Phase 2: Conceptual database design	Conceptual Schema design (DBMS-independent)		Transaction and application design (DBMS-independent)
Phase 3: Choice of DBMS			
Phase 4: Data model mapping (logical design)	Logical Schema and view design (DBMS-dependent)	Frequencies, performance constraints	
Phase 5: Physical design	Internal Schema design (DBMS-dependent)		
Phase 6: System implementation and tuning	DDL statements SDL statements		Transaction and application implementation

ER Diagrams + Mapping (1)

- Describe in words the following ERD
  - How can you identify an instance of *E*?
- Map E to relation(s)
  - What are the primary key(s)?
  - What happens to other key(s)?

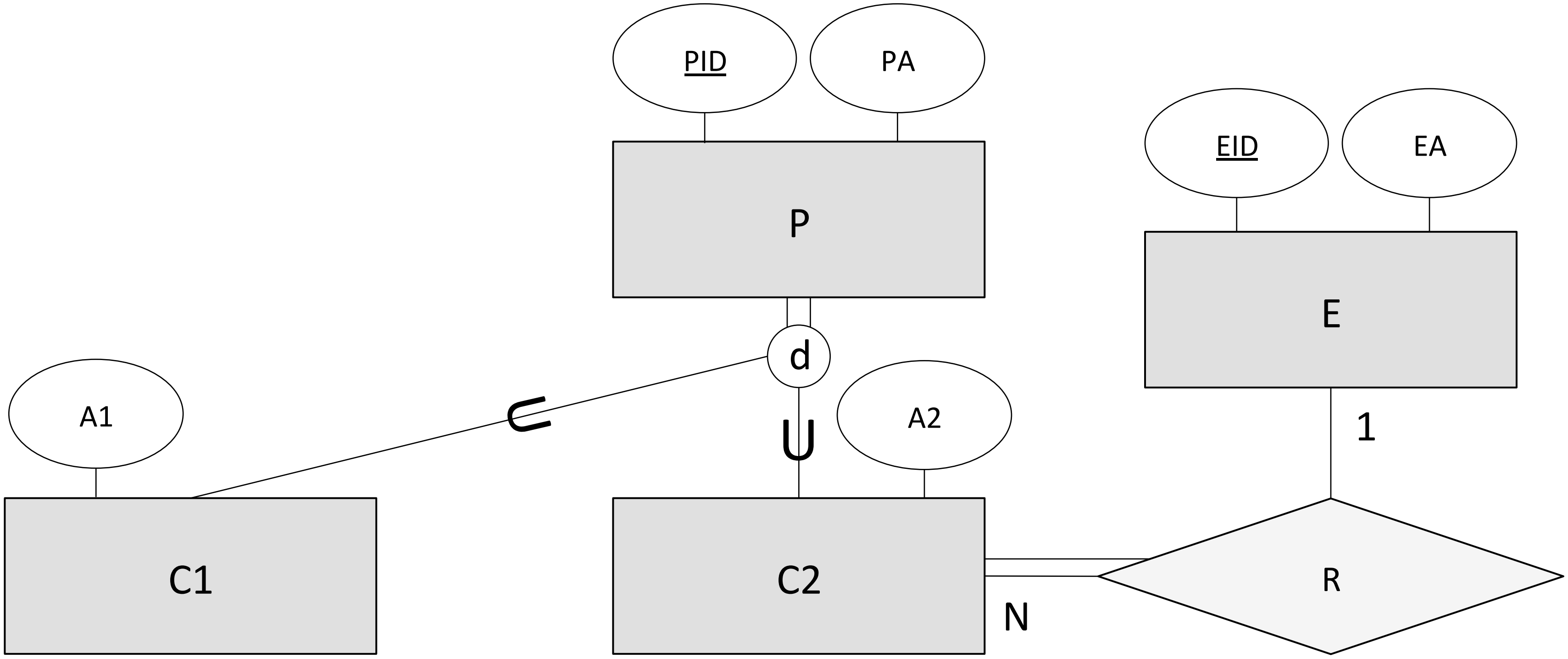


ER Diagrams + Mapping (2)

- A bank has a unique code, a name, and an address for the corporate office
- A bank can have multiple branches, each identified by a branch number and a brick-and-mortar address
- Each bank branch serves customers via two main products: accounts and loans
- An account is uniquely identified by its number, and also has a type and balance – an account can belong to one or more customers
- A loan has a unique number, as well as an amount – a loan can be made to one or more customers
- Each customer has a unique SSN, a name, as well as some number of addresses and phone numbers

ER Diagrams + Mapping (3)

Map this ERD to relations in **two** different ways. Pros and cons of each?



Relational Model

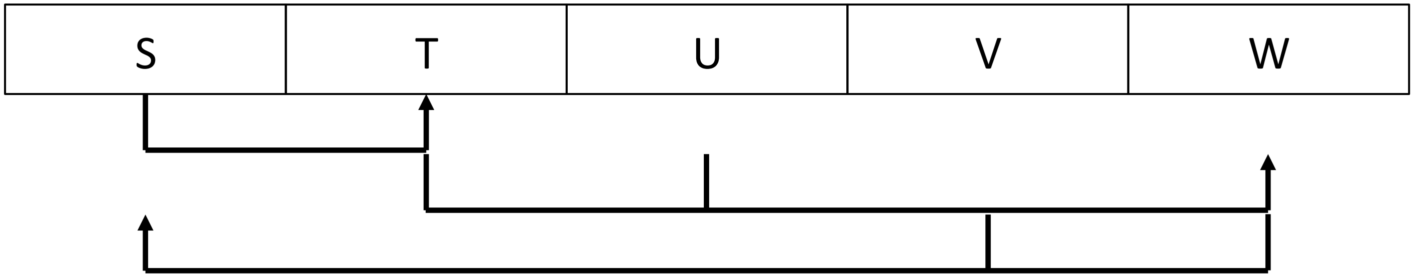
- A database is composed of?
- A table schema is composed of?
- Each [schema component] has a ____ of valid _____ values?
- What is the difference between a set vs. bag of tuples?
  - In what context does each apply?
- Provide meaning/examples of each general category of constraints: implicit, explicit, application-based, data dependencies.
- What kinds of constraints that can be defined in the schema?
  - What is a superkey vs. a key?
    - How do you identify a primary key? What happens to other super keys?
  - How do foreign keys fit in?

Normalization (1)

- What are spurious tuples?
- What is a modification anomaly? Examples: insert, update, delete
- What is an FD?
  - How is a key defined w.r.t. FD's?
- What does 2NF require? 3NF?

Normalization (2)

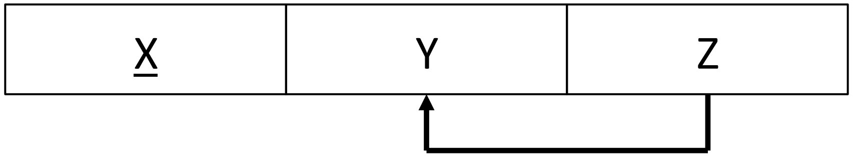
List all FDs, identify all key(s):



Normalization (3)

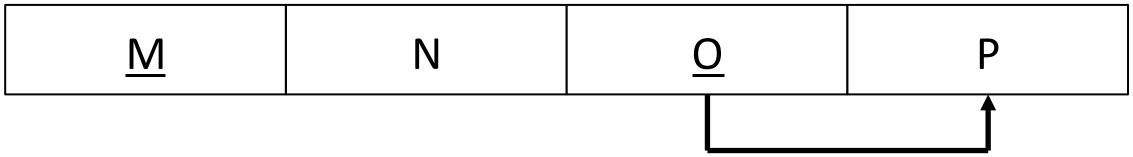
Which NF? Why? Decompose to 3NF.

Foo



X Y Z

Bar

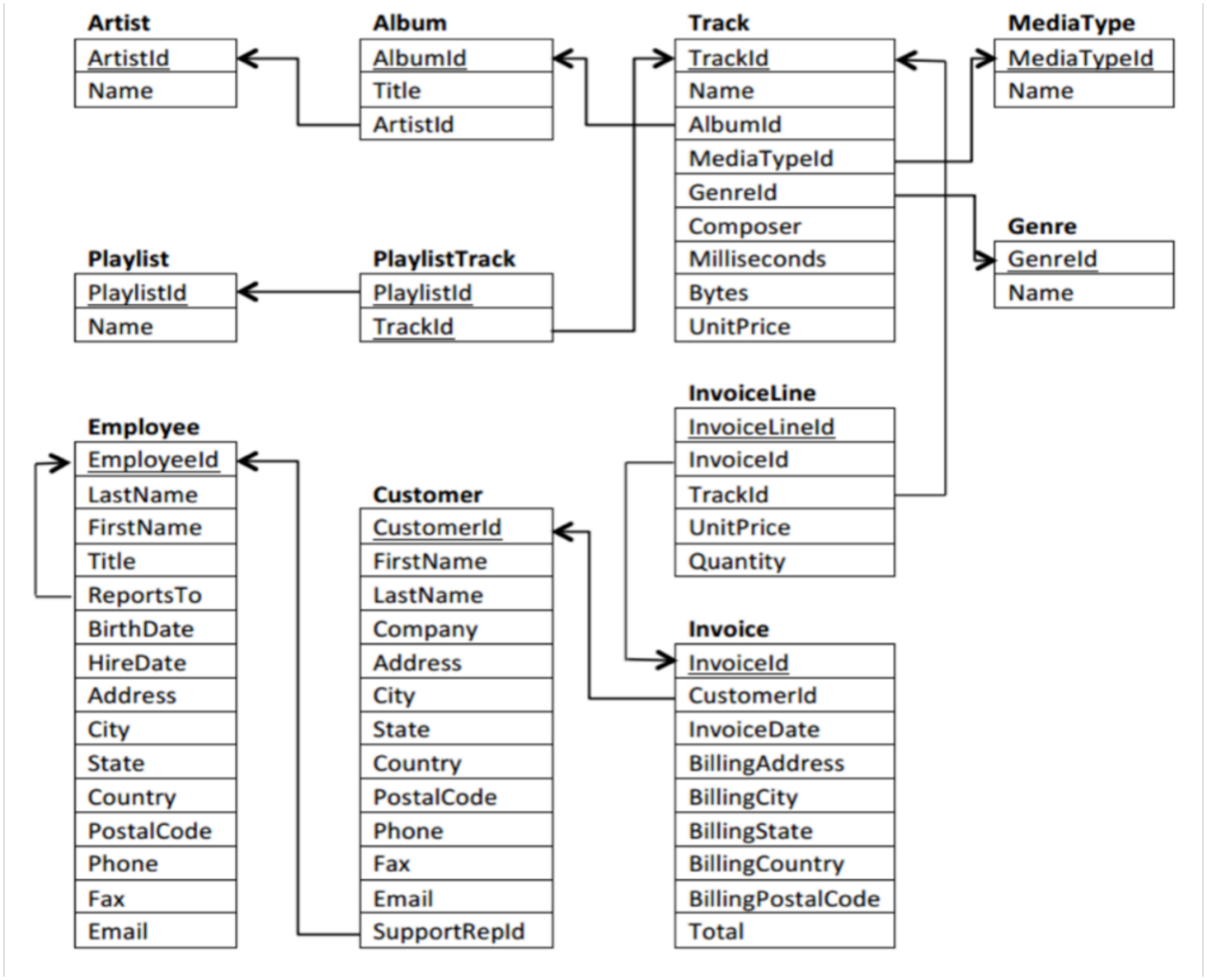


M N O P

SQL [Programming]

- Know your terms/keywords: DDL/DML, `ASSERTION/TRIGGER`, `BEGIN/COMMIT/ROLLBACK`, `GRANT/REVOKE/WITH GRANT OPTION`, `REFERENCES`, `BLOB`, ODBC, Impedance Mismatch, Result Set, Cursor
- Given a schema and a query description, write SQL
- Given SQL and a set of populated table(s), predict the result set

Chinook



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

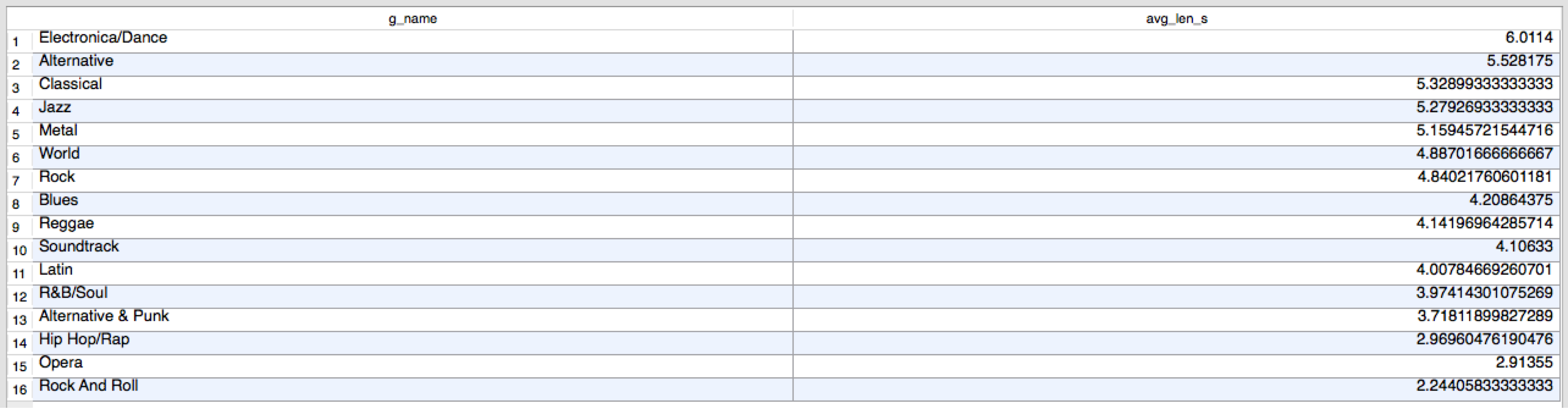
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

SQL Query (1)

For each genre, find the average length (in minutes) of each audio file in the “90’s Music” playlist. Sort the results by length (longest first), then genre name (alphabetical).



	g_name	avg_len_s
1	Electronica/Dance	6.0114
2	Alternative	5.528175
3	Classical	5.32899333333333
4	Jazz	5.27926933333333
5	Metal	5.15945721544716
6	World	4.88701666666667
7	Rock	4.84021760601181
8	Blues	4.20864375
9	Reggae	4.14196964285714
10	Soundtrack	4.10633
11	Latin	4.00784669260701
12	R&B/Soul	3.97414301075269
13	Alternative & Punk	3.71811899827289
14	Hip Hop/Rap	2.96960476190476
15	Opera	2.91355
16	Rock And Roll	2.24405833333333

SQL Query (2)

uno

a	b
1	foo
2	bar
3	baz
4	qux

dos

x	y
2	1
3	1
7	4
8	5

```
SELECT b AS var, AVG(x) AS avg_line
FROM uno LEFT JOIN dos ON uno.a=dos.y
GROUP BY b
ORDER BY b DESC
```

Physical Design/Tuning

- Terms: clustered, denormalization, [materialized] view, vertical/horizontal partitioning, EXPLAIN
- Compare B+-tree vs. Hash table index
  - When would you use one vs. other?
- Given a description of a table (number of rows, attribute cardinalities) and a query, choose the appropriate index(es) to use – justify your choice
- Given a schema and a set of queries (with frequency/execution time), develop an effective physical tuning plan (e.g. index(es), denormalization, partitioning, query rewriting)

Database Security

See/listen to all of lecture 11

- Great for terms/ideas
- Understand forms of authentication, attack, mitigation tools, inferential security, and effective methods of encryption

Inverted Index

- Given a problem description, determine if an inverted index would be applicable; if so, provide a mapping to documents/words
- Given a set of documents, produce an inverted index
- Given an inverted index, full-text query
  - Should be able to handle logical or relational form of the index (e.g. what is the SQL to find all restaurants that take reservations? and take credit cards!?)