Midterm Review

- Major Database Concepts (Chap 1, 2)
- ER Model (Chap 3)
- EER (Chap 4)
- Relational Data Model (Chap 7.1-7.3)
- Data Model Mapping
  - ER-to-Relational Mapping (9.1)
  - EER-to-Relational Mapping (9.2)
- Query Languages
  - Relational algebra (7.4-7.6)
  - SQL (Chap 8)
  - TRC (9.3)
  - DRC (9.4)

Review of Major DB Concepts

- Data and Information
  - Data: raw facts that can be recorded and have an implicit meaning.
  - Information: processed data that has value for decision making
- Database
  - A collection of data. Essential to organize data for retrieval and maintenance.
  - Characteristics of Database
- Metadata
- DBMS: Database management system
- Database system: DBMS + Data
University Database

Entities:
- students
- faculty
- courses
- offerings
- enrollments

Relationships:
- faculty teach offerings
- students enroll in offerings
- offerings made of courses

Faculty Assignment

Course Scheduling

Registration

Grade Recording

A Database System
A Simple File System

Review of Major DB Concepts (Cont.)

- Data Model
  - Definition
  - Main categories
- Database Schemas and Instances (States)
- DBMS architecture
  - Three-Schema architecture
  - Data independence
- DBMS languages: DDL, DML, SDL, VDL
Data and Its Structure

- Data is actually stored as bits, but it is difficult to work with data at this level.
- It is convenient to view data at different levels of abstraction.
- **Schema**: Description of data at some level. Each level has its own schema.
- Three schemas: physical, conceptual, and external.

Physical Data Schema

- Describes details of **how** data is stored: tracks, cylinders, indices etc.
- Early applications worked at this level - explicitly dealt with details.
- **Problem**: Routines hard-coded to deal with physical representation.
  - Changes to data structure difficult to make.
  - Application code becomes complex since it must deal with details.
  - Rapid implementation of new features impossible.
Conceptual Data Level

• Hides details.
  – In the relational model, the conceptual schema presents data as a set of tables.
• DBMS maps from conceptual to physical schema automatically.
• Physical schema can be changed without changing application:
  – DBMS must change mapping from conceptual to physical.
• Referred to as *physical data independence*.

Conceptual Data Level (con’t)

Diagram:

```
Application  Conceptual view of data
            | DBMS
            | Physical view of data
```
External Data Level

- In the relational model, the external schema also presents data as a set of relations.
- An external schema specifies a *view* of the data in terms of the conceptual level. It is tailored to the needs of a particular category of users.
  - Portions of stored data should not be seen by some users.
    - Students should not see faculty salaries.
    - Faculty should not see billing data.
  - Information that can be derived from stored data might be viewed as if it were stored.
    - GPA not stored, calculated when needed.

External Data Level (con’t)

- Application is written in terms of an external schema.
- A view is computed when accessed (not stored).
- Different external schemas can be provided to different categories of users.
- Translation from external to conceptual done automatically by DBMS at run time.
- Conceptual schema can be changed without changing application:
  - Mapping from external to conceptual must be changed.
- Referred to as *conceptual data independence*. 
Levels of Abstraction

- **View 1**: Conceptual schema
- **View 2**: Conceptual schema
- **View 3**: Physical schema
- **External schemas**: payroll, billing, records

Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.
Data indep is accomplished:

changing a schema at a lower level of DB, only the mappings between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are unchanged. Hence, the application programs need not be changed since the refer to the external schemas.
File System Critique

- File System Data Management
  - Require extensive programming in a 3GL
  - System admini. becomes difficult, as the # of file expands
  - Difficult and important to make changes in existing file structure
  - Omit security feature to safeguard data
  - Island of data information

File System Critique

- Structural and Data Dependence
  - Structural Dependence
  - Data Dependence

- Data dependence makes file system cumbersome
File System Critique

- Field Definitions and Name Conventions
  - A flexible (good) record definition anticipates reporting requirements by breaking up fields into their components.
  - Example:
    - Customer Name ⇒ Last Name, First Name, Initial
    - Customer Address ⇒ Street Address, City, State

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_LNAME</td>
<td>Customer last name</td>
</tr>
<tr>
<td>CUS_FNAME</td>
<td>Customer first name</td>
</tr>
<tr>
<td>CUS_INITIAL</td>
<td>Customer initial</td>
</tr>
<tr>
<td>CUS_AREACODE</td>
<td>Customer area code</td>
</tr>
<tr>
<td>CUS_PHONE</td>
<td>Customer phone</td>
</tr>
<tr>
<td>CUS_ADDRRE55</td>
<td>Customer street address or box number</td>
</tr>
<tr>
<td>CUS_CITY</td>
<td>Customer city</td>
</tr>
<tr>
<td>CUS_STATE</td>
<td>Customer state</td>
</tr>
</tbody>
</table>

File System Critique

- Selecting proper filed names is very important
  - descriptive within restrictions
  - Reflect documentation

- Data Redundancy
  - Data inconsistency (lack of data integrity)
  - Data anomalies
    - Modification anomalies
    - Insertion anomalies
    - Deletion anomalies
DB System Vs File System

DBMS Languages

- **Data Definition Language (DDL):**
  - Used by the DBA and database designers to specify the *conceptual schema* of a database. In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate *storage definition language (SDL)* and *view definition language (VDL)* are used to define internal and external schemas.
  - Result of compiling DDL is *catalog* (a set of tables stored in a file)

- **Data Manipulation Language (DML):** Used to specify database retrievals and updates.
  - High-level (nonprocedural, declarative) DML:
    • Describe what data is needed w/o specifying how to get it
    • DML commands (*data sublanguage*) can be *embedded* in a general-purpose programming language (*host language*), such as COBOL, PL/I or PASCAL.
  - Low-level (procedural) DML:
    • Describe what & how
    • *stand-alone* DML commands can be applied directly (*query language*).
Data Model

- **Schema**: description of data at some level (e.g., tables, attributes, constraints, domains)
- **Model**: tools and language for describing:
  - Conceptual and external schema
    - Data definition language (DDL)
  - Integrity constraints, domains (DDL)
  - Operations on data
    - Data manipulation language (DML)
  - Directives that influence the physical schema (affects performance, not semantics)
    - Storage definition language (SDL)

Database Models

- A set of logical constructs
  - Represent data structure & data relationships
- Two types of database models
  - Conceptual model
    - Logical nature of data representation
    - What is represented
  - Implementation model
    - How the data are represented
    - How the data structure are implemented
Implementation Database Models

• Hierarchical database model
• Network database model
• Relational database model

A Hierarchical Structure
Hierarchical Data Model

• Pros
  – Conceptual simplicity
  – Database security
  – Data independence
  – Database integrity
  – Efficiency dealing with a large DB

• Cons
  – Complex implementation
  – Difficult to manage
  – Lacks structural independence
  – Applications programming and use complexity
  – Implementation limitation
  – Lack of standards

• Child with multiple parents
A Network Database Model

Network Database Model

• **Pros**
  – Conceptual simplicity
  – Handles more relationship types
  – Data access flexibility
  – Data independence
  – Conformance to standards

• **Cons**
  – System complexity
  – Lack of structural independence
Relational Database Model

- **Pros**
  - Structural independence
  - Improved conceptual simplicity
  - Easier database design, implementation, mgmt, and use
  - Ad hoc query capability (SQL)
  - Powerful DBMS

- **Cons**
  - Hardware and software overhead
  - Possibility of poor design and implementation
  - Potential “islands of information” problems
Object-Oriented Database Model

• Basic Structure
  – Objects
  – Attribute
  – Class

• Comparison of OO data model and ER data model

![Object-Oriented Database Model Diagram]

OO Database Model

• Pros
  – Add semantic content
  – Visual presentation includes semantic content
  – Database integrity
  – Both structural and data independence

• Cons
  – Lack of OODM standards
  – Complex navigational data access
  – High system overhead slows transaction
Data Modeling Using the ER Model and EER

- ER Model Concepts
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
  - Relationships and Relationship Types
  - Structural constrains on relationship types (2 methods)
    - 1. Cardinality ratios, and Participation constraints (e.g., Fig 3.2, pp46)
    - 2. Specify (min, max), e.g. Fig 3.15 pp65
  - ER diagram notation (Fig 3.14 pp63)
- Relationships of Higher Degree
- Extended Entity-Relationship (EER) Model

Graphical Representation

- Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.
- E.g.
  - A program can be offered to any number of student, but every student must major in exact one program.

  \[
  \text{STUDENT} \quad \text{MajorsIn} \quad \text{PROGRAM}
  \]
  - A degree program can have many student majoring in and a student can major in at most 4 programs in the same university.
**Graphical Representation**

- Roles are edges labeled with role names (omitted if role name = name of entity set). Most attributes have been omitted.

**Relational Data Model**

- Relational Model Concepts
- Characteristics of Relations
- Relational Integrity Constraints
  - Key Constraints
  - Entity Integrity Constraints
  - Referential Integrity Constraints
- Operations
Tables

- Relational database is a collection of tables
- Heading: table name and column names
- Body: rows, occurrences of data

<table>
<thead>
<tr>
<th>Student</th>
<th>StdSSN</th>
<th>StdLastName</th>
<th>StdMajor</th>
<th>StdClass</th>
<th>StdGPA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123-45-6789</td>
<td>WELLS</td>
<td>IS</td>
<td>FR</td>
<td>3.00</td>
</tr>
<tr>
<td></td>
<td>124-56-7890</td>
<td>NORBERT</td>
<td>FIN</td>
<td>JR</td>
<td>2.70</td>
</tr>
<tr>
<td></td>
<td>234-56-7890</td>
<td>KENDALL</td>
<td>ACCT</td>
<td>JR</td>
<td>3.50</td>
</tr>
</tbody>
</table>

Relationships
Definition Summary

<table>
<thead>
<tr>
<th>Informal Terms</th>
<th>Formal Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Relation</td>
</tr>
<tr>
<td>Column</td>
<td>Attribute/Domain</td>
</tr>
<tr>
<td>Values in a column</td>
<td>Domain</td>
</tr>
<tr>
<td>Row</td>
<td>Tuple/Instance</td>
</tr>
<tr>
<td>Table Definition</td>
<td>Schema of Relation</td>
</tr>
<tr>
<td>Populated Table</td>
<td>Extension</td>
</tr>
</tbody>
</table>

Integrity Rules

- Entity integrity: primary keys
  - Each table has column(s) with unique values
  - Ensures entities are traceable
- Referential integrity: foreign keys
  - Values of a column in one table match values from a source table
  - Ensures valid references among tables
Formal Definitions I

- **Superkey**: column(s) with unique values
- **Candidate key**: minimal superkey
- **Null value**: special value meaning value unknown or inapplicable
- **Primary key**: a designated candidate key; cannot contain null values
- **Foreign key**:
  - Satisfy two conditions:
    1. $\text{Dom}(R_1.FK) = \text{Dom}(R_2.PK)$
    2. $t_1[FK] = t_2[PK]$ or NULL
- column(s) whose values must match the values in a candidate key of another table

Formal Definitions II

- **Entity integrity**
  - No two rows with the same primary key value
  - No null values in a primary key
- **Referential integrity**
  - Foreign keys must match candidate key of source table
  - Foreign keys in some cases can be null
Self-Referencing Relationships

• Foreign key that references the same table
• Represents relationships among members of the same set
• Not common but important when occurring

Review of Relational Algebra

• Domain: set of relations
• Basic operators:
  – Select (σ)
  – project (Π)
  – union (U)
  – set difference (-)
  – Cartesian product (X)
• Derived operators: set intersection, division, join
• Procedural: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression
SQL

- Data Definition in SQL
- Retrieval Queries in SQL
- Specifying Updates in SQL
- Relational Views in SQL

Summary of SQL Syntax
(PP285 Table 8.1)

SELECT <list of column expressions>
FROM <list of tables and join operations>
WHERE <list of logical expressions for rows>
GROUP BY <list of grouping columns>
HAVING <list of logical expressions for groups>
ORDER BY <list of sorting specifications>

- Expression: combination of columns, constants, operators, and functions
Join Operator

- Most databases have many tables
- Combine tables using the join operator
- Specify matching condition
  - Can be any comparison but usually =
  - PK = FK most common join condition
  - Relationship diagram useful when combining tables

Join Example

<table>
<thead>
<tr>
<th>Faculty</th>
<th>Natural Join of Offering and Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>FacSSN</td>
<td>FacName</td>
</tr>
<tr>
<td>111-11-1111</td>
<td>joe</td>
</tr>
<tr>
<td>222-22-2222</td>
<td>sue</td>
</tr>
<tr>
<td>333-33-3333</td>
<td>sara</td>
</tr>
<tr>
<td>OfferNo</td>
<td>FacSSN</td>
</tr>
<tr>
<td>1111</td>
<td>111-11-1111</td>
</tr>
<tr>
<td>2222</td>
<td>222-22-2222</td>
</tr>
<tr>
<td>3333</td>
<td>111-11-1111</td>
</tr>
</tbody>
</table>
Inexact Matching

• Match against a pattern: LIKE operator
• Use meta characters to specify patterns
  – Wildcard (* or %)
  – Any single character (?) or _)

Example
SELECT *
FROM Transcript T
WHERE T.CrsCode LIKE 'EE*'

GROUP BY
Table output by WHERE clause:
- Divide rows into groups based on subset of attributes;
- All members of a group agree on those attributes

Each group can be described by a single row in a table with attributes limited to:
- Attributes all group members share (listed in GROUP BY clause)
- Aggregates over group
GROUP BY - Example

Transcript

<table>
<thead>
<tr>
<th>1234</th>
<th>1234</th>
<th>1234</th>
<th>1234</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>3.3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Attributes:
- student’s Id
- avg grade
- number of courses

SELECT T.StudId, AVG(T.Grade), COUNT (*)
FROM Transcript T
GROUP BY T.StudId

HAVING Clause

- Eliminates unwanted groups (analogous to WHERE clause)
- HAVING condition constructed from attributes of GROUP BY list and aggregates of attributes not in list

SELECT T.StudId, AVG(T.Grade) AS CumGpa,
    COUNT (*) AS NumCrs
FROM Transcript T
WHERE T.CrsCode LIKE ‘CS%’
GROUP BY T.StudId
HAVING AVG (T.Grade) > 3.5
Example

- Output the name and address of all seniors on the Dean’s List

```sql
SELECT S.Name, S.Address
FROM Student S, Transcript T
WHERE S.StudId = T.StudId AND S.Status = 'senior'
GROUP BY S.Name, S.Address
HAVING AVG(T.Grade) > 3.5 AND SUM(T.Credit) > 90
```

Summarizing Tables

- Row summaries important for decision-making tasks
- Row summary
  - Result contains statistical (aggregate) functions
  - Conditions involve statistical functions
- SQL keywords
  - Aggregate functions in the output list
  - GROUP BY: summary columns
  - HAVING: summary conditions
SQL Summarization Rules

- Columns in SELECT and GROUP BY
  - SELECT: non aggregate and aggregate columns
  - GROUP BY: list all non aggregate columns

- WHERE versus HAVING
  - Row conditions in WHERE
  - Group conditions in HAVING

Mapping EE, EER to Relational Schema

- ER-to-Relational mapping
  - Chapter 9.1 (p290-295)
  - Lecture notes of week 8 (lec14_mapping.pdf)

- EER-to-relational Mapping
  - Chapter 9.2
  - Lecture notes of week 8 (lec14_mapping.pdf)
Relational Calculus

- TRC
  - Chapter 9.3 (p299-303)
  - Reviewed on lecture 15
  - Lecture notes of Week 8 (lec14_mapping.pdf)
- DRC
  - Chapter 9.4 (p 308)
  - Reviewed on lecture 15
  - Lecture notes of Week 8 (lec14_mapping.pdf)