cheny187
1002297034
Brian Chen
October 2017

University of Toronto, Scarborough Fall CSCC43
CSCC43 Midterm Notes

Chapter 1 – Database Fundamentals

Data – Objects or events that can be recorded on computer media
Structured Data – have a type, e.g. date, numeric, etc.
Unstructured Data – no type, multimedia (maps, documents, pictures)
Database – Organized collection of logically related data
Database Management System (DBMS) –Create, update sort or retrieve data. Enforces data integrity, assures durability of data and allows data to be shared
Information – Data that has been processed to increase user knowledge (e.g. students db gives info on gpa, #, etc.)
Metadata – Data that describes properties of end-user data and context (name, type, length, ownership, etc.)
File Systems have disadvantages over databases, namely:
Program-Data Dependence – If data is changed, all programs that use data should be altered too
Duplication of Data – Data formats may be inconsistent, compromise data integrity and wastes space
Data sharing limitations
Lengthy development times
Excessive Program maintenance
Database approach works because of
Central repository of data, managed by a controlling agent, stored in standardized convenient form
Program Data Independence, Planned data redundancy, improved data consistency, enforcement of standards, improved data quality, etc.
However databases also need, to their detriment:
New specialized personnel
Conversion costs
Constraints – A rule that cannot be violated by database users
ACID – Atomicity, Consistency, Isolation, and Durability properties of a database transaction

Chapter 2 – Data Modelling I

Relational Data Models contain three things:
Entity, a person place or object that we want to maintain data about
Entity Type corresponds to a table (Student, CourseTaken)
Entity Instance an entry into the table (100, Jane, 3.5GPA)
Strong Entity (Box) Parent entities have weak entities that depend on them – have <u>primary key</u> as well
Weak Entity (Double box) Depend on other entities, don't have primary keys, entity and identifier have double lines

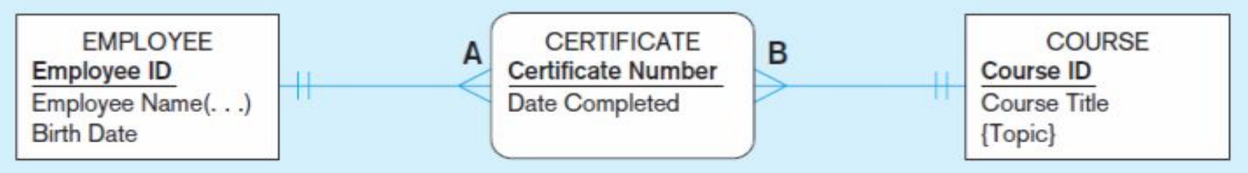
Entity Relationships describe the relationships of entities, modelled by writing relationship name on line
Relationship Instance: associations between entity instances (e.g. Jane has 3 courses, 1:M)
Relationship types: associations between entity types (students can take more than 1 course, 1:M)
Relationship degrees – Unary, Binary, Ternary
Unary – One entity related to itself (e.g. Person married to Person)
Binary – One entity related to another diff one (Employee is assigned Parking Space)
Ternary – One entity is related to two other ones (Item supplies X, Y, Z)

Attributes, a property or characteristic of an entity (e.g. ID, Name, GPA, Mark)

Business Rules are statements that define or constrain some aspect of the business, gathered by interviewing investigating and asking questions about WHO, WHAT, WHEN, WHERE, HOW, and WHY about organization **(2 per)**
Cardinality Constraint – A rule that specifies the number of instances one of an entity that can be associated with each instance of another entity (e.g. Each student at UTSC can take at most 5 courses per semester)
o = Optional, | = must, with | = one and ←- meaning many as as for relationship diagrams as well.
e.g. ----o← means optional many, ----||- means mandatory one
Attributes have different types: Required, optional, simple, composite, single values, multivalued, etc.
Composite – can have multiple meaningful components (12 Acre Heights = 12, Acre Heights)
Simple – cannot be broken down into further components (StudentID)
Multivalued – can be many (programmer, analyst, etc.)
Derived – can be calculated through related stored attribute values, represented using [ ], square brackets

Chapter 3 – Data Modelling II

Associative Entity (RoundBox) – When you have a (M;M) relationship and the relationship has an entity (e.g. Employee completes course on DateCompleted), it is an attribute of a relationship



Multiple Relationship – An entity can participate in multiple relationships (e.g. Employee supervises himself, works in department, managed by department)
Multivalued Attributes – Can also be entities, but need to have identifiers (which get funky)
Time-Dependent Data – can use a time-stamp that indicates when something occurred

Enhanced ER Diagram (EER) a model that resulted from extending ER diagrams with new modelling constructs
SuperType a parent class, like Animal o---*)--- to subtypes*
SubType, a subclass like Cat, Dog, Tiger, Bear
Relationships at SuperType level means all subtypes will participate in these things
SubType level means only subtypes will participate
Completeness Constraints – Whether an instance of a supertype must be a member of one subtype
Yes: Total Specialization Rule, goes ==o--)-----
No: Partial Specialization Rule, goes –o--)-----
Disjoint Constraints - Supertypes may simultaneously be a member of two or more subtypes
Disjoint Rule – Instance of supertype can only be ONE subtype **(d)** (patient must be outpatient or resident)
Overlap Rule – Can be multiple subtypes **(o)** (a machine part may be a purchased and a manufactured one)
Subtype Discriminator – An attribute of supertype whose values determine the target subtype(s)
Disjoint – simple attribute that describes it (e.g. Employee Type = 'H', 'S', 'C')
Overlapping – composite attribute that describes multiple (e.g. Part type = (MP))

Chapter 4 – Database Design I

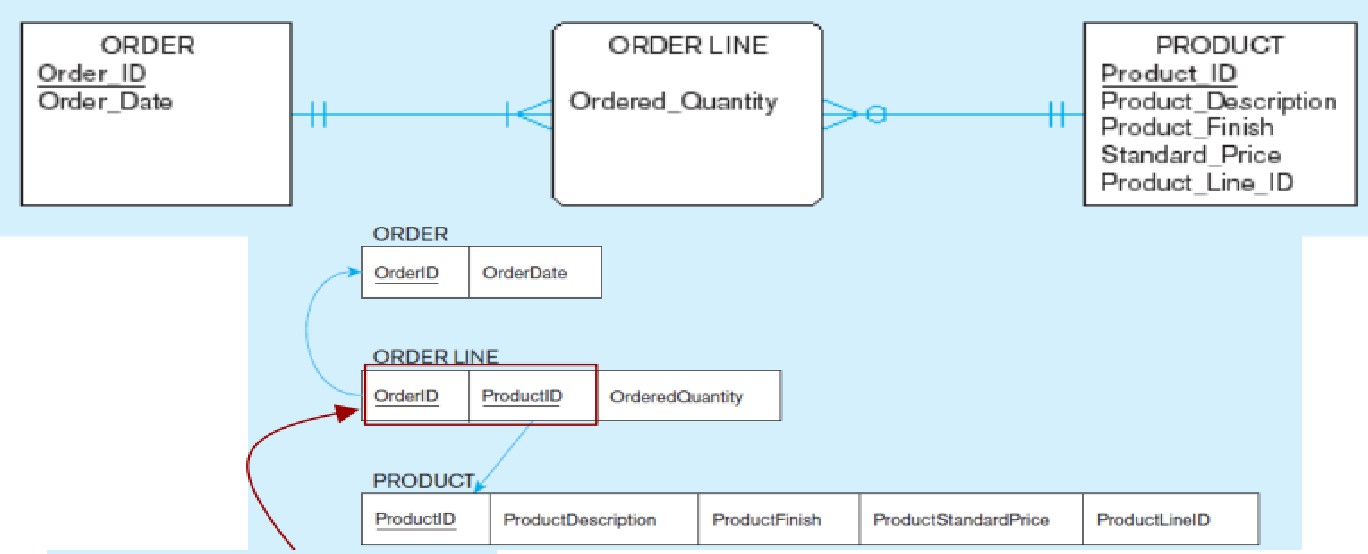
All Relations are Tables, but not all **Tables** are **Relations**
Requirements for Table to be Relation:
It must have a unique name
Every attribute value must be atomic (no composite or multivalued)
Every row must be unique (cant have repeated rows)
Attributes/columns must have unique names
Order of columns must be irrelevant
Order of rows must be irrelevant
Primary key – an attribute or combination of attributes that uniquely identifies each row in a relation (e.g. ID#)
Composite Key – a primary key that contains more than one attribute
Foreign Key – an identifier that enables a dependent relation to refer to its parent relation
Schema – shorthand of a collection of relations, (e.g. Customer = CustomerID, CustomerName, eg....)
Data Integrity are the constraints that assures the accuracy and integrity of data in the database
Domain Constraint all values that appear in a column must originate from same domain (e.g. size, type)
Entity Integrity each entity must have a primary !NULL key
Entity Integrity Rule – no primary key or component of a primary key can be null
Referential Integrity in a 1:M relationship, any foreign key value must match a primary key value in the relation of the one side
Transforming EER into Relations

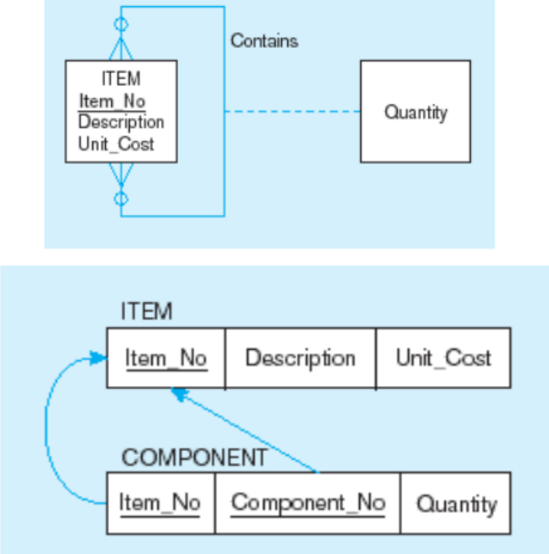
1. Map regular entities
2. Map weak entities
3. Map binary relationships
4. Map associative entities
5. Map unary relationships
6. Map n-ary relationships
7. Map supertype/subtype relationships

We choose to make **multivalued attributes** into dependent relations with foreign keys
Mandatory dependent things must satisfy the referential integrity rule

Chapter 5 – Database Design II

Mapping Associative Entities



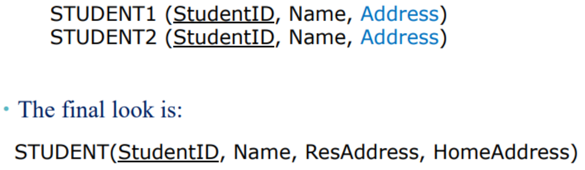


← Mapping Unary Relations

Normalization the process of decomposing relations with anomalies to produce smaller well-structured relations
Well Structured Relations
No Data Redundancy
No Insertion Anomaly adding new roles should not force user to insert irrelevant data
No Deletion Anomaly deleting rows should not cause loss of data that would be needed for future rows
No Modification Anomaly changing data should not force changes to other rows due to duplication
Functional Dependency the value of one attribute (determinant) determines the value of another (ID -> Name)
Candidate Key – An attribute or combination of attributes that uniquely identifies a row in a relation (candidate keys will become primary keys).
Nonredundancy – no attribute in key can be deleted without destroying the property of unique identification

Chapter 6 – Database Design III

Depends on the Key, The Whole Key, and Nothing But the Key, so help me Codd
FIRST NORMAL FORM – No multivalued attributes, every attribute value is atomic
SECOND NORMAL FORM – In 1NF, No partial functional dependency, every non-key depends on the entire primary key
THIRD NORMAL FORM – In 2NF, and nothing have dependencies on non-PK attributes
Boyce-Codd – If it is in 3NF and every determinant is a candidate key (basis for 3NF)
Top Down Analysis – From Business Rule -> ER -> Relation
Bottom Up Analysis – Merging the relations
Synonym – Two attributes that have the same meaning but diff names
Alias – An alternative name used for an attribute


STUDENT1 (StudentID, Name, Address)
STUDENT2 (StudentID, Name, Address)

- The final look is:

STUDENT(StudentID, Name, ResAddress, HomeAddress)

Homonym – An attribute that can have more than one meaning (e.g. account can be saving or current or loan account) **[ABOVE]**
Transitive Dependencies – When two 3NF merge, there may be transitive dependencies (i.e. one item may depend on another)
Enterprise Key – A unique key across all platforms
VarChar – Variable character, can change storage space required dynamically
DMBS support: Default value, Range control, Null value control, Referential integrity

Indexing using B-Trees – Better storage and faster access

- When >100 values, <30 values
- Index search fields
- Avoid indexing long values
- Can't index null values

Chapter 7 – SQL I

Schema – The structure that contains descriptions of objects created by users (tables, views, constraints, etc.)

Catalog – A set of schemas that constitute the description of a database

Data Definition Language (DDL) – Physical design & Maintenance

- Create, Alter, Drop
- Tables, views, and indexes
- Establishing constraint

Data Manipulation Language (DML) - Implementation

- Update, insert, modify and query a database

Data Control Language (DCL) – Implementation and Maintenance

- Grant or revoke privilege

DDL – CREATE TABLE, CREATE VIEW, CREATE SCHEMA

Default Values and Domain Constriction

ProductFinish VARCHAR(20) Default 'walnut'
CHECK (ProductFinish IN ("a" … "z")),…

Key Setting

CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));

Chapter 8 – SQL II

DML – Insert, etc.

Inserting

INSERT INTO table_name column_name etc.

Delete

DELETE FROM table_name WHERE condition

Update

UPDATE table SET col=… WHERE …

Indexing

CREATE INDEX index_name ON table_name
DROP INDEX index_name ON table_name

SELECT

SELECT - List columns
FROM – from where data is obtained
WHERE – conditions under which a row is included

Aliases cannot be used in Where clause

GROUP BY – categorization of results
HAVING – indicate conditions under which a group will be included
ORDER BY – sorted

Alias

SELECT P.PRODUCTID as PROD ← ALIAS

Wild Cards –

* - All
% - Like (collection of character)
_ - exactly one character
Null

Aggregate Functions – AVG< SUM, COUNT, MAX, MIN, ROUND, LOWER

Scalar Aggregate – Single value returned from SQL Query with aggregate function

Vector Aggregate – Multiple values returned from SQL query with aggregate function

Views – Provide users controlled access to tables

Dynamic Views – Virtual table created dynamically, instead data from base table displayed

Materialized View – Copy or replication of data, must be refreshed periodically to match new data

CREATE VIEW EXPENSIVE_STUFF_V AS SELECT PRODUCT_ID, PRODUCT_NAME, UNIT_PRICE FROM PRODUCT_T WHERE UNIT_PRICE >300 WITH CHECK_OPTION;

Pros – Simplified query commands, assists with security, programming productivity, little storage space, customized)

Cons – Processing time each time view is referenced, may not directly be updatable

Chapter 9 – SQL III

Relational Algebra - Union, Intersection, Difference, Selection, Projection, Cartesian product, joins

Cartesian Product – R x S = {(a, b) | a E R && b E S}

Natural Join/Equi Join - R ⋈ S – Join on some same ID, no repetition

Outer Join – A join that includes all tuples even the ones that do not have matching values

Left Outer Join – R ⟕ S

Right Outer Join – R ⟖ S

Full outer join – R ⟗ S

SubQuery –

```
SELECT CustomerName FROM Customer_T WHERE CustomerID IN
	(SELECT DISTINCT CustomerID from Order_T)
```

Correlated subquery - Is a subquery (a query nested inside another query) that uses values from the outer query.

COALESCE – Returns first non-null expression among arguments

NULLIF – Returns null if both are equal, otherwise first argument

CASE –

```
SELECT ProductNumber, Category =
CASE ProductLine
	WHEN 'R' THEN 'Road'
	WHEN 'M' THEN 'Mountain'
	WHEN 'T' THEN 'Touring'
	WHEN 'S' THEN 'Other sale items'
ELSE 'Not for sale'
END,
Name
FROM Production.Product
ORDER BY ProductNumber;
```

Chapter 9 – SQL IV

Transaction – A set of commands that is atomic, All or None

To Ensure Transaction Integrity –

Begin Transaction – Starts

Commit – Makes all updates permanent

Rollback – Cancels updates since last commit

BEGIN transaction

….do some sql stuff

END transaction

Routines – Program modules that execute on demand

Functions, procedures, triggers

Routine – Call procedure_name(params) -> does code -> does database

CREATE PROCEDURE sp_name ([proc_parameter[,...]])
routine_body

CREATE FUNCTION sp_name ([func_parameter[,...]]) RETURNS type
routine_body

Trigger – Insert/Update/Delete -> triggers -> does database

CREATE TRIGGER trigger_name { BEFORE | AFTER } { INSERT | UPDATE | DELETE } ON tbl_name FOR EACH ROW
trigger_body

Interactive SQL – What we've done ti lnow

Embedded SQL – Including hard-coded statements into program written in C/Java/Python

Embedding SQL in 3GL (3rd generation language)

- More flexible and accessible
- Performance improvement

- Grants access to only app not users

Dynamic SQL – Ability for program to generate SQL code on the fly

Chapter 10 – SQL Injections

SQLi – SQL injection

- Sends malicious code to server
- Database runs code as legitimate code
- Data is revealed
- Works on dynamic portion of application

Can be used to:

- Delete and modify data
- Run OS commands
- Create DOS attacks

Tautology – Creates a condition that always evaluates to true

Inferential –

Illegal/logically incorrect queries

- Collects constraints

Blind SQL Injection –

Content based

- E.g. targets content

Time-based

- E.g. sleep(t)

SQLi Attack Countermeasures –

◦ **Sanitize data**

◦ **White list input validation**

◦ **Use of parameterized queries**

Not parameterized: query = ″SELECT name FROM my_table
WHERE id = ‘” + id + ”‘ ”

Parameterized: query = ″SELECT name FROM My_table
WHERE id = ? ”

◦ **Use of stored procedures**