

The relational model and basic aspects of SQL

Michael Emmerich and Paul Breukel
Leiden Institute for Advanced Computer Science, Leiden
University

February 4, 2011



Introduction to SQL

- ▶ Overview of SQL and the Relational Model
 - ▶ Data Definition Language (DDL)
 - ▶ Data Manipulation Language (DML)
- ▶ Logical Database Design: Translating ER Diagrams to SQL-DDL
- ▶ Advanced Queries in SQL-DML

The Relational Model: Why studying it?

- ▶ Most widely used model. Vendors: IBM, Microsoft, Oracle, Sybase, Informix etc.
- ▶ Recent competitor: object-oriented model
 - ▶ ObjectStore, Versant, Ontos
 - ▶ A synthesis emerging: object-relational model: Informix Universal Server, UniSQL, Oracle, UDB

The relational model

- ▶ Relational database: a set of relations
- ▶ Relation: made up of 2 parts:
 - ▶ Instance : a table, with rows and columns.
 - ▶ Number of Rows = *cardinality*
 - ▶ Number of fields = degree / arity.
 - ▶ Schema: specifies name of relation, plus name and type of each column.
 - ▶ e.g. Students(sid: string, name: string, login: string, age: integer, gpa: real).
 - ▶ Can think of a relation as a set of rows or tuples (i.e., all rows are distinct).
- ▶ Example: Relation with degree of 5 and cardinality of 3:

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Do all rows have to be distinct?

Creating a table (relation) in SQL:

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

```
CREATE TABLE Students (
```

```
  sid:CHAR(20),
```

```
  name:CHAR(20),
```

```
  login:CHAR(10),
```

```
  age:INTEGER,
```

```
  gpa:REAL)
```

Creates the Students relation. Observe that the *type* (*domain*) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

Destroying and altering a table in SQL:

`DROP TABLE Students`

- ▶ Destroys the relation Students. The schema information and the tuples are deleted.

`ALTER TABLE Students ADD COLUMN firstYear:
integer`

- ▶ Adds column firstYear to table.

`ALTER TABLE Students DROP COLUMN firstYear:
integer`

- ▶ Drops column

Inserting and deleting rows:

- ▶ Insert a row in table Students

```
INSERT INTO
```

```
  Students (sid, name, login, age, gpa)
```

```
  VALUES (53688, Smith, smith@ee, 18, 3.2)
```

- ▶ Delete a row in table Students

```
DELETE FROM Students S WHERE S.name = Smith
```

Queries in the relational model: example

- ▶ Given

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- ▶ "Get IDs and login of all students who are 18 years old" translates to:

SELECT * FROM Students WHERE age = 18

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

Example for querying multiple tables

- ▶ Given the two relations Enrolled and Students:

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

What does the following query compute?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Queries in the relational model

- ▶ A major strength of the relational model: supports simple, powerful querying of data
- ▶ Queries can be written intuitively, and the DBMS is responsible for efficient evaluation
 - ▶ Precise semantics for relational queries.
 - ▶ Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.
 - ▶ Allows schema adaptation and extensions
 - ▶ Intuitive referencing (by *names of columns* instead of position numbers)

Relational Model: Summary

- ▶ Information is stored in tables
- ▶ Implemented in SQL
 - ▶ Data definition language: Definition of Tables (schema), insertion of rows
 - ▶ Data manipulation language: Updating tables, querying information

What is logical database design?

- ▶ In logical database design we solve the problem of how to arrange data by means of tables
- ▶ Many constructs of the ER Diagram can be translated directly into a conceptual model of a DB
- ▶ SQL allows to handle some additional types of constraints that cannot be expressed in the ER Diagram
- ▶ The main difficulty: How to enforce integrity constraints, referential integrity, avoid redundancy?

Integrity constraints

- ▶ Integrity constraints are introduced to check semantical correctness of the data entered by the user, e.g. key constraints, participation constraints, numerical conditions
- ▶ There are different possibilities to enforce integrity constraints:
 - ▶ Many integrity constraints can be enforced by means of check statements in SQL that make sure that certain logical conditions on the data are satisfied
 - ▶ Often the enforcement of an integrity constraint can be achieved by an appropriate design of the conceptual schema. If this is possible, it is usually preferable to the introduction of check constraints.
 - ▶ Not all semantical constraints can be enforced. There is always some responsibility on the user, who enters the data.

Referential integrity

- ▶ Referential integrity is a concept discussed in different contexts: e.g. programming with pointers, hypertext documents, and databases
- ▶ Referential integrity is given if all pointers (references) point to existing objects (e.g. no dead links in the internet), and existing objects are accessible from somewhere
- ▶ In databases this plays a role when designing tables that encode relationships

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Question: is referential integrity applied in the tables above?

Logical Database Design from ER Diagrams

- ▶ Given: An ER Diagram (and some additional constraints that may not be represented in ER Diagram)
- ▶ Needed: A description of tables (conceptual schema) and some information of how to deal with updating and deletion

Basic SQL Statements for logical database design

- ▶ `CREATE TABLE tableName (columnName1 dataType, columnName2 dataType,)`
- ▶ Within `CREATE TABLE` we will learn to use
 - ▶ `PRIMARY KEY`, `UNIQUE`: To indicate (candidate) keys of tables
 - ▶ `FOREIGN KEY`: To make references to other tables
 - ▶ `ON DELETE`, `ON UPDATE`, `NOT NULL`: To ensure integrity

Translating simple entity sets and keys



Constraint: ISBN is also unique

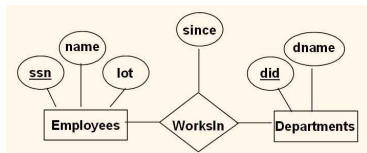
- ▶ Translating the entity set book to a SQL relation:
- ▶ `CREATE TABLE Books (author CHAR(20), title CHAR(100), publisher CHAR(20), year INTEGER, PRIMARY KEY (author, title))`
- ▶ A table has one and only one primary key
- ▶ There can be more than one attribute (set) that has to be unique. This constraint can be expressed by candidate keys (UNIQUE):
- ▶ `CREATE TABLE Book (author CHAR(20), title CHAR(100), publisher CHAR(20), year INTEGER, PRIMARY KEY (author, title), UNIQUE (isbn))`

Relationship sets to tables

- ▶ Different approaches for different constraint types
 - ▶ Many-to-many relationships
 - ▶ Many-to-one (or one-to-one) relationships
 - ▶ Many-to-one relationships with total participation
 - ▶ Weak dependencies
 - ▶ Dealing with "IS-A" constructions ...

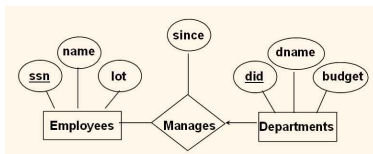
Translating a many-to-many relationship set

- ▶ Two tables for the entity sets and one table for the relationship set



```
CREATE TABLE WorksIn(  
    ssn CHAR(1), did INTEGER, since DATE,  
    PRIMARY KEY (ssn, did),  
    FOREIGN KEY (ssn) REFERENCES Employees,  
    FOREIGN KEY (did) REFERENCES Departments)
```

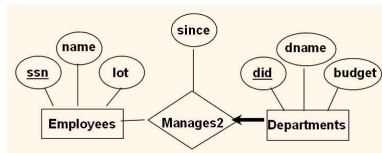
Translating a many-to-one relationship set



```
CREATE TABLE Manages(
    ssn CHAR(1), did INTEGER, since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

- ▶ Note that did is now the only key attribute
- ▶ Alternative: CREATE TABLE Departments (....., FOREIGN KEY (ssn) REFERENCES Employees)

Translating participation constraints in many-to-one relationships

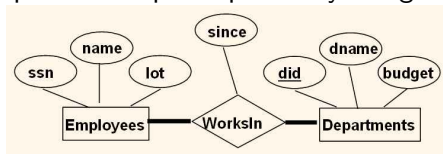


```
CREATE TABLE Departments (
    did INTEGER, dname CHAR(20), budget REAL, ssn
    CHAR(11), since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees)
```

- ▶ As each department has a unique manager, we can combine Manages and Departments

Total participation and many to many relationships

We may capture total participation by using NOT NULL



```
CREATE TABLE Departments(  
    budget REAL, dname CHAR(20), did INTEGER,  
    ssn INTEGER NOT NULL,  
    FOREIGN KEY (ssn) REFERENCES Employees,  
    ON DELETE NO ACTION)
```

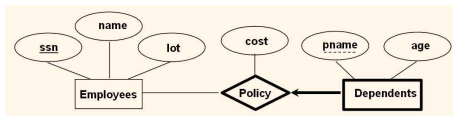
- ▶ After deletion of an employee the constraint must still be satisfied
- ▶ Conclusion: We need to use a CHECK statement (later introduced) to enforce this constraint

One-to-one correspondence

- ▶ Total participation on both sides, and one-to-one relation
 - ▶ both tables can be combined, for example:

```
CREATE TABLE PairDancers(  
    dancerId INTEGER,  
    partnerId INTEGER,  
    dname CHAR(20),  
    birthday DATE,  
    PRIMARY KEY (dancerId),  
    FOREIGN KEY (partnerId) REFERENCES PairDancers,  
    ON DELETE CASCADE ON UPDATE CASCADE)
```

Weak dependencies



```
CREATE TABLE Dependents(
    pname CHAR(20), age INTEGER, cost REAL,
    ssn CHAR(11) NOT NULL,
    PRIMARY KEY (pname, ssn), FOREIGN KEY (ssn)
REFERENCES Employees, ON DELETE CASCADE ON UPDATE
CASCADE)
```

- ▶ When the owner of the weak dependent is deleted, the dependent also will be deleted to maintain referential integrity.
- ▶ When the primary key of the owner is changed, the foreign key of all the dependents will be changed as well, to remain referential integrity.