



Relational Database Concepts

IBM Information Management Cloud Computing Center of Competence
IBM Canada Labs

1

© 2011 IBM Corporation

In this presentation you will learn about basic relational database concepts

Agenda

- Overview
- Information and Data Models
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
 - Normalization

Supporting reading material & videos

- **Reading materials**

- Database Fundamentals eBook
 - Chapter 1: Databases and information models
 - Chapter 2: The relational data model
 - Chapter 3: The conceptual data model (optional)
 - Chapter 4: Relational database design (optional)

- **Videos**

- db2university.com course AA001EN
 - Lesson 1: Relational database concepts

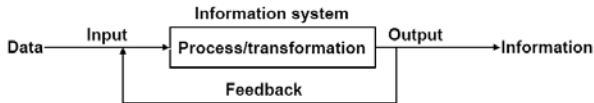
Agenda



- **Overview**

- Information and Data Models
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
 - Normalization

Data vs. Information



- **Data:** Collection of letters, numbers or facts
- **Information:** Processed data that provides value

Let's first understand the difference between the concepts of "data" versus "information". As the figure shows, **data** is a collection of letters, numbers or facts, which by itself derive little or no value. When data is passed as input to an *Information System* to be processed and transformed, it becomes **information**. Information provides value. During the processing of the data, feedback can be passed back to fine tune the type of data that is collected and how it is collected.

Databases and DBMS

- **Databases**
 - A repository of data
- **DBMS (Database management system)**
 - Software system that manages databases
 - The terms “Database”, “DBMS”, “data server”, “database server” often used interchangeably to refer to a DBMS
- **Why a DBMS?**
 - Security
 - Can handle many users with good performance
 - Allows for concurrency while keeping data consistent
 - Protects from disaster

05/24/2011

Template Documentation

6

© 2011 IBM Corporation


Data can be stored in many ways, for example, text files, spreadsheets, etc. However, using these methods pose problems when trying to access or manipulate the data. For example, if 100 users want to access and make changes to a text file, there will be problems opening the file. This is why databases are needed. Databases are not necessarily cheaper than other solutions, but are better to manage data.

A **Database** is a repository of data, and a **DBMS (Database management system)** is the software system that manages databases. Often, the terms “Database”, “DBMS”, “data server”, “database server” are used interchangeably to refer to a DBMS

Why a DBMS?

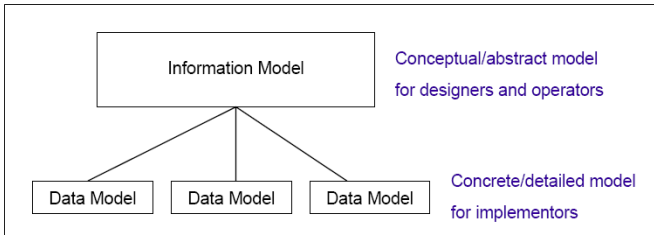
A DBMS offers security, good performance even when thousands of users are working with it, concurrency while keeping data consistent, and protection from disaster.

Agenda

- Overview
-  • **Information and Data Models**
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
 - Normalization

Information and Data Models

Relationship between an Information Model and a Data Model



An information model is an abstract, formal representation of entities that includes their properties, relationships and the operations that can be performed on them.

The main purpose of an Information Model is to model managed objects at a conceptual level, independent of any specific implementations or protocols used to transport the data.

Data Models, on the other hand, are defined at a more concrete level and include many details. They are intended for software developers and include protocol-specific constructs. A data model is the blueprint of any database system.

The figure illustrates the relationship between an Information Model and a Data Model.

Data Models

- Network
- Hierarchical
- Relational
- Entity-Relationship
- Extended relational
- Semantic
- Object-oriented
- Object-relational
- Semi-structured

Data model proposals can be split into nine historical epochs:

Network, Hierarchical, Relational, Entity-Relationship, Extended relational, Semantic, Object-oriented, Object-relational, Semi-structured

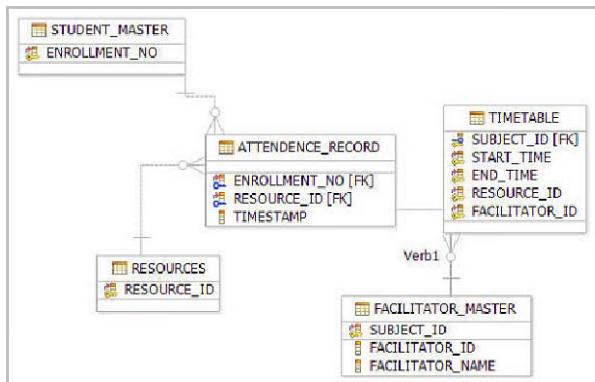
We will focus in this training in the "Relational Data Model" only.

Agenda

- Overview
- Information and Data Models
- **The relational model**
 - **Entity-Relationship diagrams**
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
 - Normalization



Relational Model



The relational data model is simple and elegant. It has a solid mathematic foundation based on sets theory and predicate calculus and is the most used data model for databases today.

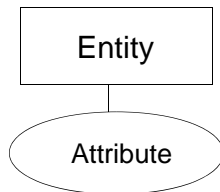
This figure illustrates an example showing an Entity-Relationship i.e., an E-R diagram that represents entities or tables and their relationships for a sample relational model.

Entity-Relationship Diagrams

- Building Blocks

- Entities

- Attributes



Entity-relationship diagrams (ERDs) are very popular in support of the relational model.

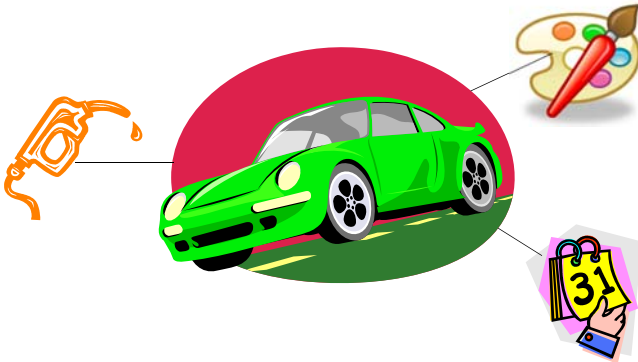
The building blocks of an ERD are Entities and Attributes.

An entity is drawn as rectangles and attributes are drawn as ovals as can be seen in this chart.

Attributes are connected with a line to exactly one entity.

An entity is an object that has some properties.

Entity and Attributes

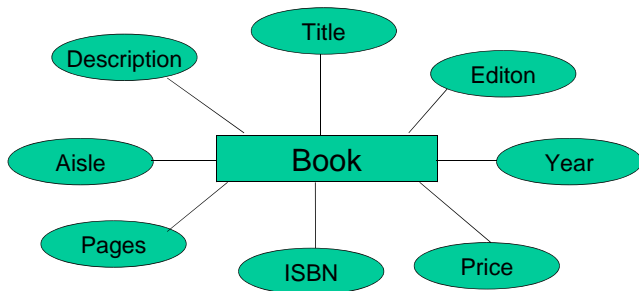


An entity can be a noun. For example, a car.

Attributes are certain properties of that entity.

For example, a car comes in different colors, have different fuel types like petrol or diesel and a make, the year in which it was made.

ER diagram



Let us take another example of a book. A book is an entity here.

Say, a book has certain properties like title, edition, year of publication, price, ISBN, # of pages, etc. These properties are nothing but the attributes.

Now, you can see that we have got one entity, a book represented in a rectangle. The rectangle and several attributes drawn in ovals connected to this one entity.

This pictorial representation is the ER diagram which is abstract and is just the beginning of many things that go in the actual design of a relational database model.

Exercise: Identify entities and attributes

House

Phone #

Social Security Number

Computer

Product

Date

Height

Order #

Can you identify the entities and attributes from the following list?

Did you get them right?

House

Phone #

Social Security Number

Computer

Product

Date

Height

Order #

House, date, computer, social security number, height, order #, product, phone #.

House, computer, product are entities.

It is easy to identify entities as they are objects.

Rest all is some kind of properties that can only be associated with a particular object. Let us recall that an entity is represented with a rectangle and an attribute with an oval.

Agenda

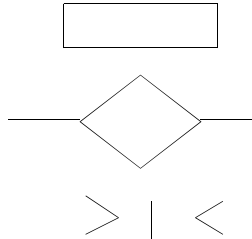
- Overview
- Entity-relationship diagrams
- The relational model
 - Entity-Relationship diagrams
 - **Types of relationships**
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
 - Normalization



Relationships

• Building Blocks

- Entity sets
- Relationship sets
- Crows Foot notations



Building blocks of a relationship are entity sets, relationship sets, and crows foot notations.

You are already familiar with the representation of entity sets in rectangles.

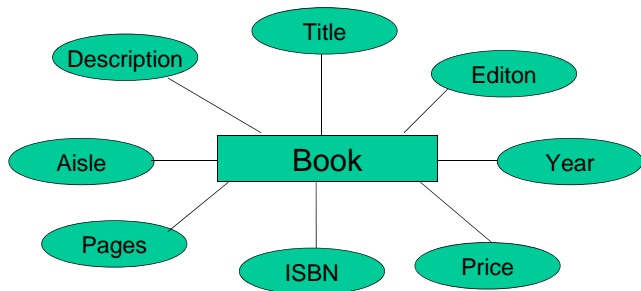
Relationship sets are represented by a diamond with lines connecting associated entities.

There are different techniques employed in representing relationships.

We shall use the crows foot notations for ease of understanding.

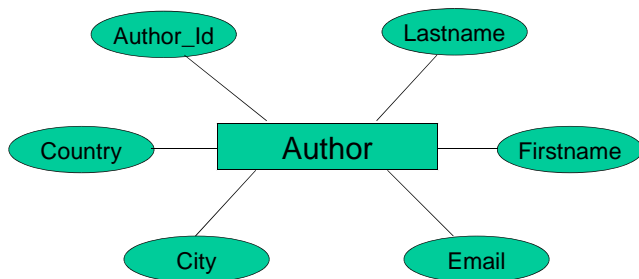
Some of these are greater than symbol, less than symbol and a vertical line.

ERD of Book



You may recall that we drew an ERD for the entity Book like this.

ERD of Author

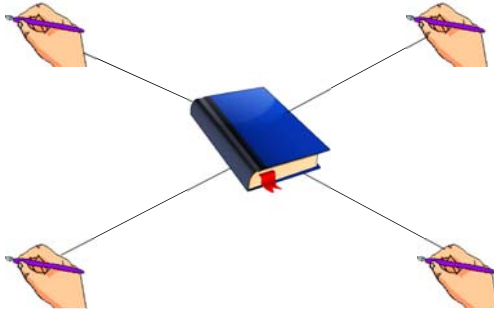


Let us take one more entity Author whose ERD would look like this.

The entity author has attributes like lastname, firstname, email, city, country and author ID.

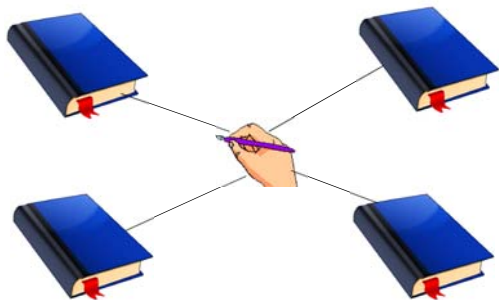
Let us see how these entities "Book" and "Author" relates to each other.

Example 1



We shall go over an example. We all know a book must be written at least by one author. It is also common for two authors to pen down a book. At times, we also see many authors coming together to write a book.

Example 2



Let us take another example. An author can write just one book, or two books or multiple books. In all these cases, there is a relationship between the book and the author.

Types of Relationships



One-to-one Relationship

We now move on to defining the types of relationships that exists between entities.

We saw that a book needs at least an author.

The relationship set that comes here is authored-by. We can say that one book must be authored by one author.

The thick lines indicate each entity in the entity set is involved in at least one and exactly one relationship. This is termed as "one-to-one relationship".

You might have noticed that we are using only entities here. Attributes are often omitted as they can clutter the relationship diagrams.

Types of Relationships (Continued)



One-to-many Relationships

Earlier we saw, more than one author can write a book. This can be represented with a different crow's foot notation. In this case, a less than symbol.

This indicates, that one book entity is participating in more than one relationship in the relationship set. This is one-to-many relationships. We can also term this as many-to-one relationships wherein we say many authors authoring a single book.

Types of Relationships (Continued)



Many-to-many Relationships

Let us check on how to represent when we have many authors authoring many books.

We are going to use greater than and less than symbols on either side of the relationship set.

This is many-to-many relationship where in each entity in the entity set is participating in more than one relationship.

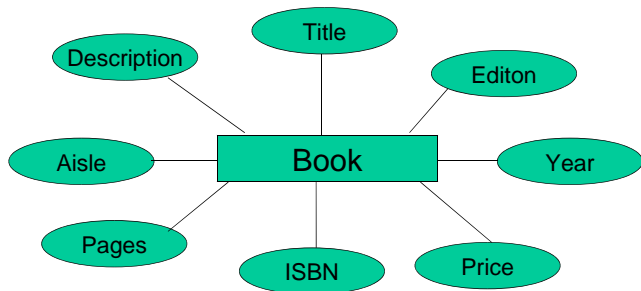
This is many books being authored by many authors, or many authors authoring many books.

Agenda

- Overview
- Entity-relationship diagrams
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - **Mapping entities to tables**
 - Relational model concepts
 - Relational model constraints
 - Normalization



ERD revisited

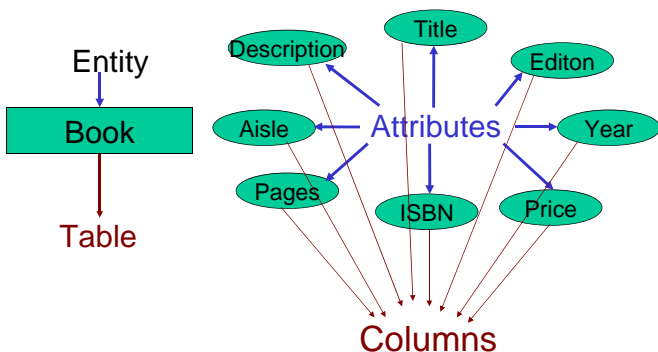


Let's now see how entities are mapped into tables. As you know, ERDs are the basic foundation for designing a database.

In a relational database design, we first begin with ERDs and later map them to the tables in a database.

You are quite familiar with this ERD aren't you? Entity Book having several attributes. Let us see how we map these entities and attributes to a table.

Mapping entity to a table



For ease of understanding, let me separate out the entity from the attributes.

Book is the entity and these are the attributes associated with the entity.

While mapping, the entities translates into tables. In this case, entity "book" becomes a table bearing the same name Book.

All the attributes translates into columns in a table. We shall now see how a table would be represented in a Relational database model.

Mapping entity to a table (Continued)

Table: Book

Title	Edition	Year	Price	ISBN	Pages	Aisle	Description
Database Fundamentals	1	2010	24.99	978-0-98662 83-1-1	300	DB-A02	Teaches you the fundamentals of databases
Getting started with DB2 Express-C	1	2010	24.99	978-0-98662 83-5-1	280	DB-A01	Teaches you the essentials of DB2 using DB2 Express-C, the free version of DB2

You all know that a table is a combination of rows and columns. While mapping, an entity becomes the table. Having said that, it has not yet taken the form of rows and columns.

Attributes which gets translated to columns in a table provides the actual table form.

We can later add some data values to each of these columns which completes the table form.

Mapping entity to a table (Continued)

Table: Author

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

Take a look at another entity author which is deciphered as a table from the ERD we had earlier.

All the attributes make up the columns and some data values into the columns will complete the table here.

Agenda

- Overview
- Entity-relationship diagrams
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - **Relational model concepts**
 - Relational model constraints
 - Normalization



Relational Model Concepts

Dr. E.F. Codd of IBM in 1970:

“A Relational Model for Large Shared Data Banks”

- Building Blocks
 - Relation
 - Sets

Let's now talk about Relational Model Concepts. The model was first proposed by Dr. E.F. Codd of IBM in 1970 in his paper was titled “A Relational Model for Large Shared Data Banks”.

The building blocks of Relational model are **Relation** and **Sets**.

Relational Model of data is based on the concept of “Relation” and a relation is a Mathematical concept based on the idea of “sets”.

A set is an unordered collection of distinct elements. It is a collection of items of same type.

It would generally have no order and no duplicates.

The relational data model was designed to provide better data independence

A Relational Database

- Relational Database

- Relation

- Relation Schema

- Relation Instance

A relational database is a set of relations.

A relation is the mathematical term for a table.

A **table** in turn is a combination of **rows** and **columns**.

A relation is made up of 2 parts – namely Relation Schema and Relation Instance.

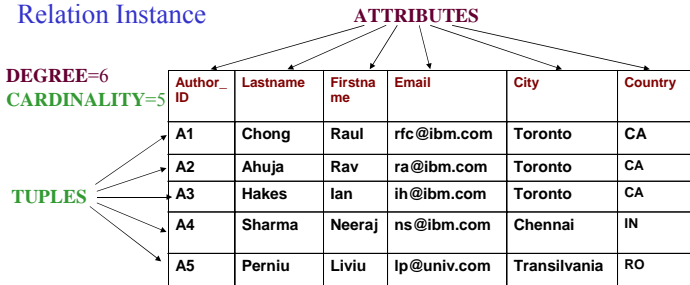
A Relation schema specifies the name of a relation, name and type of each of the columns with its attributes.

A Relation

AUTHOR(Author_ID: char, lastname: varchar, firstname: varchar, email: varchar, city: varchar, country: char)

Relation Schema

Relation Instance



A DOMAIN is the set of all possible values for a specific attribute

We shall refer to the entity Author we have seen in our previous lessons.

AUTHOR is the name of the relation. Author_ID is an attribute which can hold data of type char. Likewise lastname, firstname, email and city have type of varchar. If you are wondering what these are, they are **data types**. The last attribute Country is of char data type.

This constitutes the relation schema.

A Relation instance on the other hand, is a table made up of rows and columns.

The **columns** are termed as **attributes** or **fields** in a relation and the **rows**, **tuples**.

Degree is a term referred to the number of columns in a relation

and **Cardinality** is the number of tuples.

The chart shows the degree as 6 since we have 6 columns and cardinality of 5 as we have 5 data rows.

A Domain is the set of all possible values for a specific attribute. For example, for the attribute "Country" it would be the set of all possible country codes

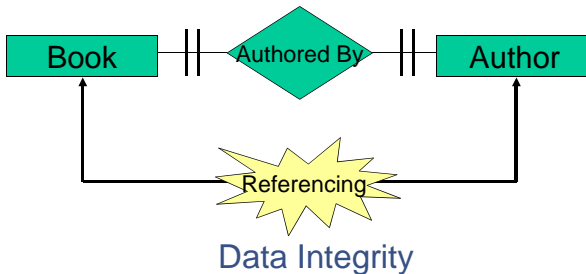
Agenda

- Overview
- Entity-relationship diagrams
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - **Relational model constraints**
 - Normalization



Relational Model Constraints

Business Rules



Let's provide an overview of Relational Model Constraints. Within any business, data must often adhere to certain restrictions or rules.

Let us take our Book and Author example.

Unless a book is written by an author, the book virtually does not exist.

At least a one-to-one relationship is a must.

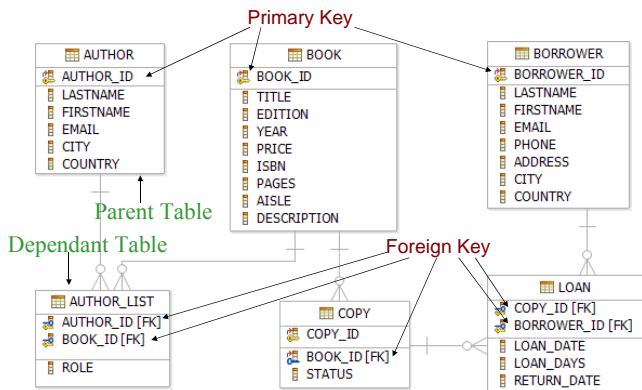
As you can see in the example, either entity book refers to Author entity to look up the author information or visa versa.

This in relational data model is termed as referencing.

This is very crucial for establishing data integrity between two relations in relational databases.

We can have multiple referencing and not necessarily be limited to just one-to-one relationship.

ERD representation of a Relational Data Model



37

© 2011 IBM Corporation

Take a look at the Relational model representation in ERD form.

We have several entities like Author, Book, and Borrower etc. Also note Author_ID, BOOK_ID and BORROWER_ID carry a special icon that features a key.

Such attributes are termed as Primary Keys. Observe the entities at the lower portion of this ERD. Some attributes have FK mentioned next to them in brackets. These are termed as Foreign Keys.

Notice these entities are part of the relationship set between the entities that are above them. All of them have a one to many relationship established between them.

A **primary key** of a relational table is a set of columns that uniquely identifies each record in the table. A primary key is recommended but **optional**.

Foreign key is a set of columns referring to a primary key of another table. It is optional, but used to establish referential/data integrity

Before moving on to the relational model constraints, let us learn two more terminologies that are often used in a relational model.

A table containing a primary key that is related to at least one foreign key is known as a **Parent table**. In our example, Author entity is the parent table. Book is also a parent table.

A table containing one or more foreign keys is known as a **dependent table**. In this diagram, author_list has two foreign keys that refer two different parent tables.

Constraints

- Entity Integrity Constraint
- Referential Integrity Constraint
- Semantic Integrity Constraint
- Domain Constraint
- Null Constraint
- Check Constraint

Back to constraints, this help implement the business rules, we have the following constraints defined in a relational database model.

Entity integrity constraint, referential integrity constraint, semantic integrity constraint, domain constraint, null constraint, and check constraint.

Let us learn these in detail in the following slides.

Entity Integrity Constraint

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

To identify each tuple in a relation, the relation must have a primary key. The **primary key** is a unique value that identifies each row. This is the entity integrity constraint. Generally the terms primary key constraint and unique constraints are used. To implement these constraints, indexes are used which will be covered in detail in later presentations.

Entity integrity constraint says that no attribute participating in the primary key of a relation is allowed to accept null values.

For explanation purpose, I shall take the example of the relation Author we have already seen in our previous presentations.

Entity Integrity Constraint

↓

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

As you know, Author_ID is the primary key for this relation Author.

Now this key identifies each tuple in the relation.

Suppose I say Author_ID A1, it points to the author Raul Chong from Toronto.

Entity Integrity Constraint

↓

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
NULL	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

What if I replace the value A1 with NULL? We can still identify the author Raul Chong.

Entity Integrity Constraint

↓

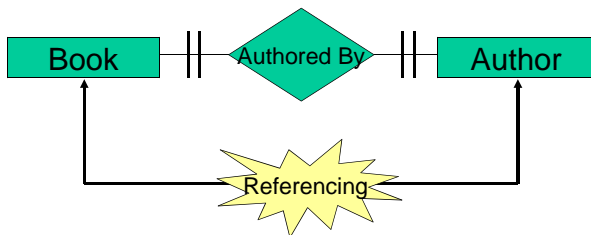
AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
NULL	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
NULL	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

What if I replace A4 also with NULL? Now, which NULL would identify which row? Isn't that difficult?

Hence, this constraint, Entity integrity constraints emphasizes on having no null values for the attributes playing the primary key role.

Referential Integrity Constraint



Referential integrity constraint defines relationships between tables and ensures that these relationships remain valid.

As mentioned in the very beginning of this presentation, for a book to exist, it has to be written by an author. Else, there is no meaning for just a book to exist all by itself.

Foreign keys or triggers in relational databases help implement this referential integrity constraint.

Semantic Integrity Constraint

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

Semantic integrity constraint

For this, let us see another example in the author relation.

Semantic Integrity Constraint

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	12(*)&^23	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

If the column city contains some junk value like this instead of Toronto, does it give any meaning to the column name city?

A semantic integrity constraint refers to the correctness of the meaning of the data.

Domain Constraint

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

A domain constraint specifies the permissible values for a given attribute.

Let us take the example of country attribute in our relation author.

We know that this particular attribute must contain a two-lettered country code such as CA for Canada, IN for India etc.

Domain Constraint

AUTHOR



Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	34
A2	Ahuja	Rav	ra@ibm.com	Toronto	34
A3	Hakes	Ian	ih@ibm.com	Toronto	34
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

Supposing a value of 34 is entered for the country, does it make any sense? No right?
Of course unless there is another table which maps Country codes in letters to that of numbers. But we do not want to induce redundancy either.

NULL Constraint

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

NULL constraints specify that attribute values cannot be null.

NULL Constraint

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	NULL	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	NULL	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

Let us see what happens if in the author relation a null value for either lastname or firstname attributes is entered.

It poses difficulty in identifying the right author just in case, either the lastname or firstname is same. This means, an author must have a lastname and a firstname and not a NULL.

Check Constraint

BOOK

Title	Edition	Year	Price	ISBN	Pages	Aisle	Description
Database Fundamentals	1	2010	24.99	978-0-9866283-1-1	300	DB-A02	Teaches you the fundamentals of databases
Getting started with DB2 Express-C	1	2010	24.99	978-0-9866283-5-1	280	DB-A01	Teaches you the essentials of DB2 using DB2 Express-C, the free version of DB2

CHECK constraints enforce domain integrity by limiting the values that are accepted by a column.

Author relation does not have a suitable attribute where-in the check constraint can be explained easily. So I will take the example of Book relation.

Check Constraint



BOOK

Title	Edition	Year	Price	ISBN	Pages	Aisle	Description
Database Fundamentals	1	2010	24.99	978-0-98662 83-1-1	300	DB-A02	Teaches you the fundamentals of databases
Getting started with DB2 Express-C	1	2015	24.99	978-0-98662 83-5-1	280	DB-A01	Teaches you the essentials of DB2 using DB2 Express-C, the free version of DB2

The year attribute is the year in which a particular book is published.

Would it be meaningful to have a year greater than the current year 2011?

Agenda

- Overview
- Entity-relationship diagrams
- The relational model
 - Entity-Relationship diagrams
 - Types of relationships
 - Mapping entities to tables
 - Relational model concepts
 - Relational model constraints
- **Normalization**



Normalization

- **Process in database design to remove redundancies**
- **Example:**

Consider the following table listing all the tasks of an employee:

ID	Name	Office City	Extension	Task
1	John S	Toronto	54213	Planning
1	John S	Toronto	54213	Marketing
1	John S	Toronto	54213	Testing
2	Susan P	New York	59867	Marketing
3	Jennifer L	Chicago	59415	Planning
3	Jennifer L	Chicago	59415	Testing

Problem:

If John moves to a new city, all entries related to John must be updated

05/24/2011

Template Documentation

53

© 2011 IBM Corporation

Normalization is used to reduce redundancies. There are various forms of normalization, such as the 3NF (3rd Normal Form), BCNF (Boyce-Codd Normal Form) and more. Each form represents a different degree in which the data must be constrained. This also allows you to be able to determine if you do not have enough relationships to sufficiently connect all of the keys in the schema.

There are possibilities that a set of data cannot reach a certain form because there is insufficient ability to create constraints for that set.

In this example, you can see, if you need to make a slight change to the table, all of the records that are linked, must be changed as well. For example, if John moves out of the Toronto Office, you need to make the change three times.

In the subsequent slide, we will show you how normalization will help you reduce the redundancies in your tables.

Normalization (continued)

No redundancy, no anomalies, no loss of information

Employee Table

ID	Name	Office City	Extension
1	John S	Toronto	54213
2	Susan P	New York	59867
3	Jennifer L	Chicago	59415

Task Table

ID	Task
1	Planning
2	Marketing
3	Testing

Employee Tasks Table

Employee ID	Task ID
1	1
1	2
1	3
2	2
3	1
3	3

05/24/2011

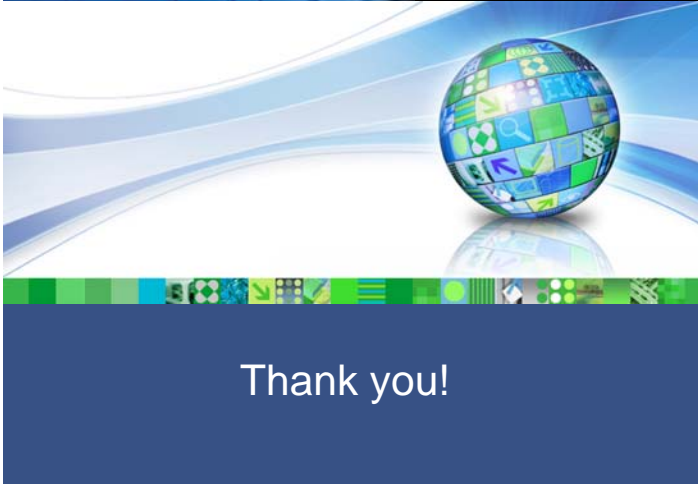
Template Documentation

54

© 2011 IBM Corporation

Here in this example, you see that the larger table has been broken up into smaller tables with less fields; however, each of the smaller tables has a foreign key in the larger table. That foreign key associates with a primary key in the smaller tables which can then be joined together to recreate the whole selection as the larger table.

In this form you can see that all of the values still properly co-exist and that if you make a change somewhere you do not need to make as many changes due to the reduction in redundancies. For example, if John moves out of the Toronto office, only 1 update is required now.



Use the forum in the db2university.com course AA001EN if you have technical questions about the materials covered in this course. Fellow students, faculty and IBMers can help you!