

Relational Database Model

Basic Structure

Relation/ Table

Named

No repeating fields

Columns

named attributes

must be atomic values

values valid within a domain

Rows

also called tuples

similar to record

must have primary key

Keys

Super key: attribute(s) that identify a tuple. always at least one. relation can have more than one

Candidate key: min set of attributes that uniquely identify a tuple. minimal superkey. may be more than one.

Primary key: one per relation only. chosen candidate key

Examples

Employee (Emp-ID, Emp-Name, Emp-Birthdate, Emp-Address, Emp-Salary)

■ - Super key

■ - Candidate key

■ - Primary key

Employee-Project (Emp-ID, Project-ID, Emp-Title-Proj, Hours-Worked)

Foreign Key

used to reference another relation

Terminology

Domain - set of atomic valid values of one or more attribute. may be specified as a data type

Atomic Values - indivisible data values

Null Value - designates a missing value

Degree - number of attributes (columns) in a relation

Cardinality - number of rows in a relation

Intension - schema of a relation

Extension - data (tuples) in a relation

Characteristics of a Relation

- 1) ordering of rows w/in a relation
no particular order. ~~no~~ unordered set
- 2) ordering of attributes w/in a relation
no particular order as long as attribute and value is maintained

Relational Constraints

Domain/Integrity Constraints

specify the valid values of each attribute

Entity Integrity Constraints

states that no attribute of a primary key can contain a null value

Referential Integrity Constraint

a foreign key can contain a valid value of the primary key in the same relation OR contain a null value

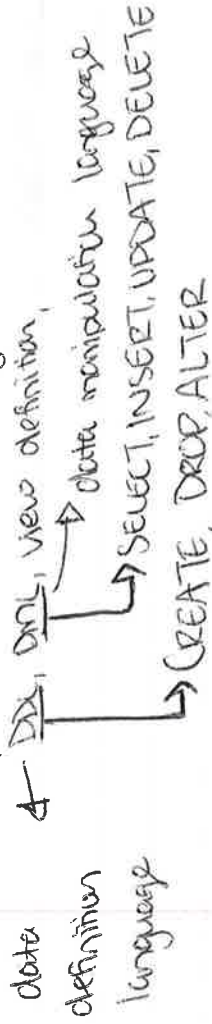
Relational Operators

Update operators - insert, delete, modify

Retrieval operators - relational algebra, relational calculus, SQL

SQL (structured query language)

comprehensive database language



Normalization to 3NF

Redundancy

when attribute values are repeated unnecessarily

Anomalies

caused by redundant info. must find all copies of info in order to prevent inconsistencies when updating

Deletion Anomaly

occurs when data is lost during a deletion that we still want

Insertion Anomaly

occurs when we cannot insert some info into a row

because of a violation

Functional Dependency

constraints in the data that depend upon whether or not two

tuples agree on certain components

Person (SSN, Age, Gender)

FD = $\{ \text{SSN}_1 \rightarrow \text{Age}_1 \}$

$\text{SSN} \rightarrow \text{Gender}$

$\text{Age} \rightarrow \text{Gender}$

Since two people of the same age can be of different genders

Normal values

1NF (First Normal Form)

all values are atomic

2NF (Second Normal Form)

prime attribute - attribute that appears in a key of a relation

non-prime attribute - attribute that

doesn't appear in a key of a relation

in 2NF if it's in 1NF and each ~~non-prime~~ non-prime attributes

are fully dependent on PK

3NF (Third Normal Form)

in 3NF if it's in 2NF and none of its non-prime attributes are

transitively dependent on its key

↳ np attribute is " — upon the pk of a relation

if there is also a np attribute that functionally determines the attribute

SAL

Create Table Syntax
CREATE TABLE <table-name> (
 <attribute> <type> → name of attribute in table to be identified
 [not null|unique][pk] char(N), varchar(N),
 {, <attribute> <type>
 [not null|unique][pk]}
 [, <pk constraint>]
 [{, <foreign key constraint>}];

Examples

CREATE TABLE person (

ssn char(9) primary key,
 fname char(10) not null,
 lname char(10) not null,
 phone# char(10);

CREATE TABLE student (

ssn char(9),
 classification char(6),
 gpa decimal(4,3),
 total_hours integer,
 primary key (ssn),
 foreign key (ssn) references person (ssn);

ALTER TABLE SYNTAX

ALTER TABLE <table-name> add (

<attribute> <type>

{, <attribute> <type>});

ALTER TABLE <table-name> modify (

<attribute> ~~<type>~~ <new-length>
 {, <attribute> <new-length>});

Example

```
alter table person add (  
  birthday char(8);  
alter table section modify (  
  title char(25),  
  description char(50));
```

Drop Table Syntax

```
Drop Table <table-name>  
[cascade constraints];
```

Example

```
Drop Table person  
cascade constraints;
```

Insert Syntax

```
insert into <table-name>  
  values (<value-list>);  
insert into <table-name>  
  (<attribute-list>  
  values (<value-list>);  
insert into <table-name>  
  select * from <another-table-name>;
```

Example

```
insert into student  
  values ('283mi', 'senior', 2.294, 100);
```

Delete Syntax

```
delete from <table-name>  
[where <condition>;]
```

Update Syntax Example

```
update student  
  set classification = 'senior'  
  where total_hours > 60;
```